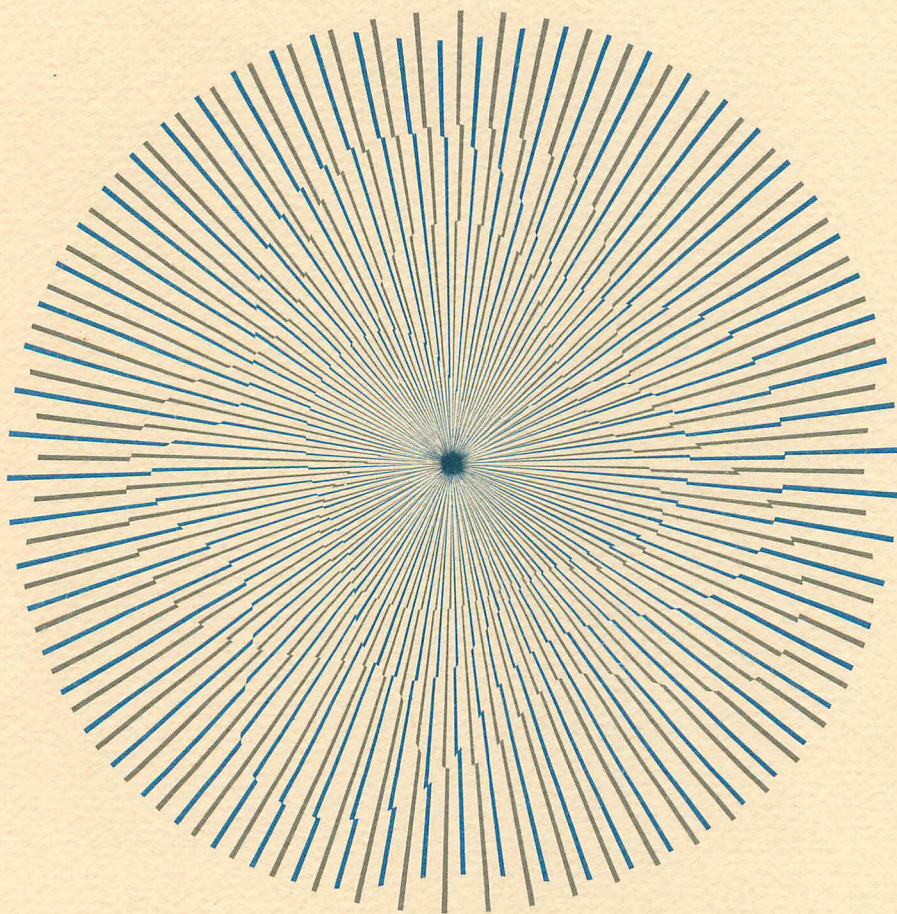
DIGITAL COMPUTER THEORY

電子計算機の基礎

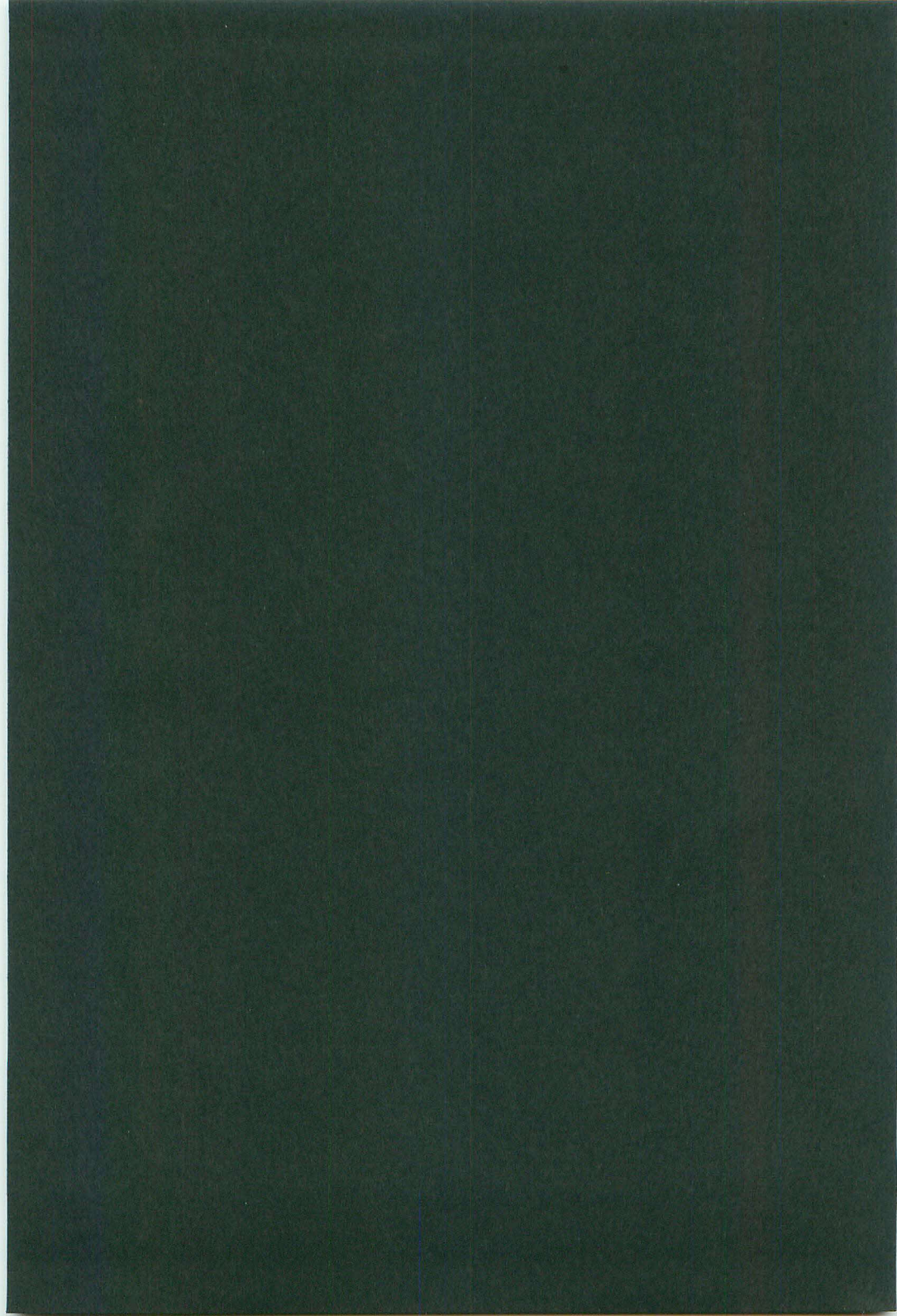
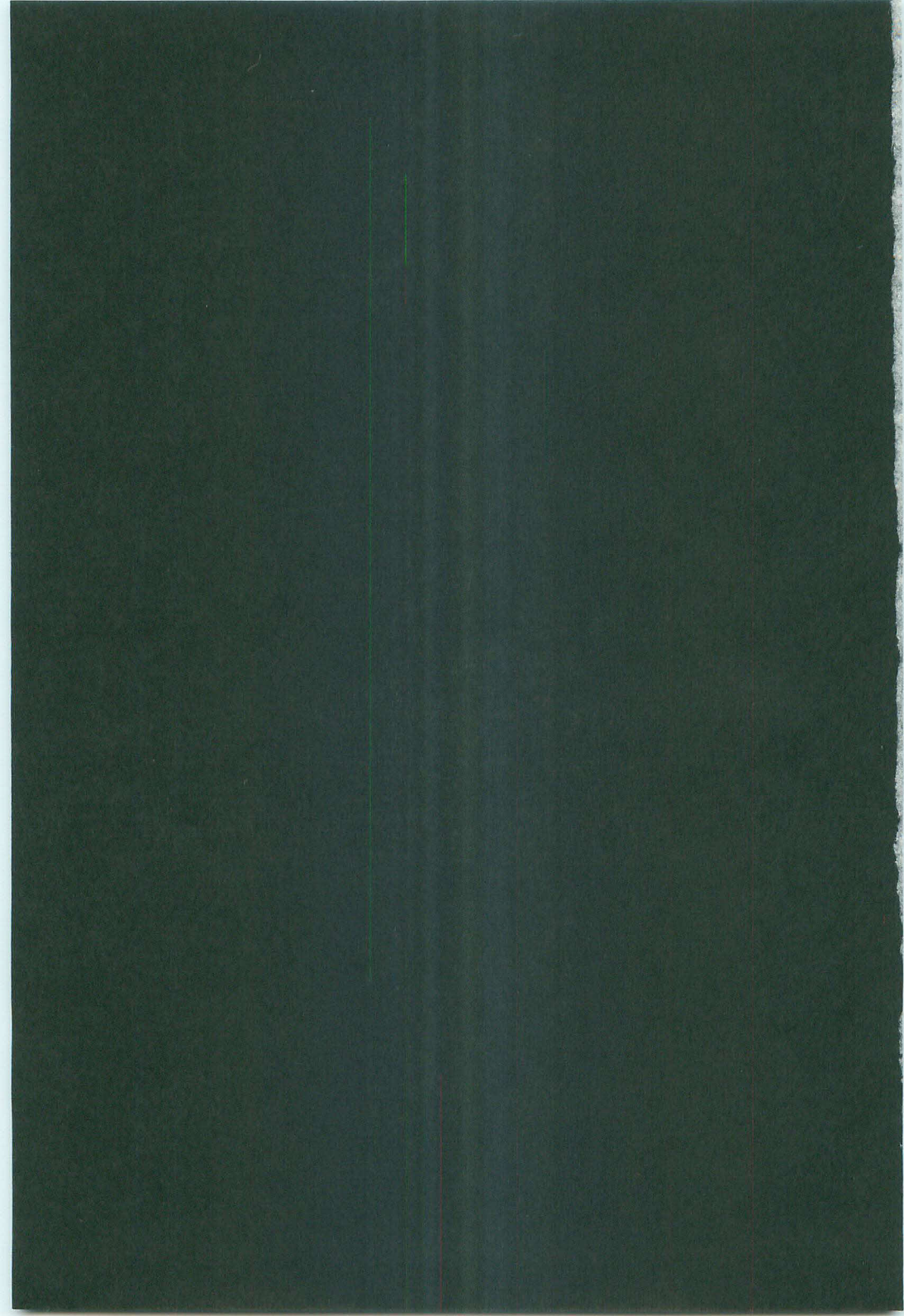
プログラマのためのハードウェア

L.ナシェルスキー著

慶応義塾大学教授　理学博士　浦　昭二　慶応義塾大学教授　工学博士　北川　節　共訳

培風館

本書は大学教養程度のテキストとして書かれたもので，計算機自体に関する基礎知識として必要十分な材料を収録してある。計算機ハードウエア関係の書物は一般に電気工学的に難解に過ぎるか，あるいは概論的，通俗的に過ぎたものが多いが，本書はこの点できわめて中庸をえた好著である。

内容は基礎，電子計算機回路，諸装置の3部から構成され，基礎編では主に電子計算機に用いられる数学と機械語プログラムについて説く。ここから出発して各部は巧みに関連づけられ，数字が機械的・電気的にいかに変換されていくかの回路を述べ，その回路が組み合わされて実際にどのような装置として使用されているかを順を追って解説する。本文には内容のチェックをかねた多数の例題や問題を適宜挿入し，各章末には要約と問題を収め，その章での総括を行なっている。多くの写真や図解を用いている点も読者の理解を助け，興味を増すであろう。

なお，予備知識はほとんど必要でなく，大学(理工系) 2～3年生，プログラマおよび会社・研究所等の初級技術者にとって好適な参考書・テキストである。

DIGITAL COMPUTER THEORY

電子計算機の基礎

プログラマのためのハードウエア

L.ナシェルスキー著

慶応義塾大学教授　理学博士 浦　昭二　慶応義塾大学教授　工学博士 北川　節　共訳

培風館

Louis Nashelsky

DIGITAL COMPUTER THEORY

装幀／永井一正＋大竹雄介

訳者序

最近の計算機の利用の伸びはまことに驚異的なものがある。計算機を利用するには FORTRAN や COBOL など，プログラミング言語の知識を持てばすむのであるが，その場合でも，計算機そのものの知識をある程度備えていることが望ましい。また，訳者が文科系の大学生や高校生に接触した経験からみて，彼らは単に計算機を使えるようになるということのほかに，計算機がどのように作られ，どのような特性を持っているかについて正しい知識を持ちたがっているように思えた。

計算機に関する書物がたくさん出回っている割には，上のような目的に適した書物は少ないように思われる。この本は決して電子計算機の専門書として高度のものとはいえないが，計算機の構成全般にわたって，懇切な解説がしてあり，入門書として最適のものといえよう。

著者には今年の5月ニューヨークで会ったことがあるが，彼の属する大学はニューヨーク市立の工科系短期大学で，豊かな教育施設を備えており，教員が熱心に教育に打ち込んでいる姿が感じられた。

最後になったが，本書の翻訳にあたっては，中西正和君（慶大大学院生），大野瑞夫君（慶大修士，現三越勤務），および吉田輝夫君（三菱電機勤務）の甚大な協力を得たことを記して，感謝のしるしとしたい。日本語版のために，多くの写真を日本アイ・ビー・エム株式会社，東京芝浦電機株式会社，および高千穂交易株式会社から快く提供していただいた。また培風館編集部の方々には多大の労をわずらわせた。ここに記して感謝の意を表したい。

1968 年 10 月

浦　昭二
北川　節

日本語版への序

多くの大学やそこの学生達は，計数型計算機とその始まりから熱心に取り組んできている。1940 年代後半から 50 年代のはじめにかけて，最初の近代的計算機が作られて以来，大学では，計算機そのものについて，次には計算機プログラミングについて教育してきた。1950 年頃からは，その主体は純粋な数学者や技術者の領域から，事務家や社会科学者などへ移っている。学部学生の教育は今や教育のあらゆる分野で計算機研究と強い関係を持っている。

大学院，大学学部，短期大学，高校，はては小学校の水準までもの教育について徹底的な検討がされている。これらの課程で学ぶ基本的な主題はまったく似通ったものである。この本は大学，短期大学，技術学校，高校などでそれぞれ時間数や強調の度合をかえて使用することができる。

この本では，まず第一に計数型計算機の一般領域を扱い，小型で容量も小さく，価格もやすい計算機が，非常に大型の計算機と，根本的には同じように構成され，組織されていることを示している。計数型計算機の一般的利用および特殊用と汎用の計算機の間の相違に言及している。

2章では，どのように計算機が最も直接的な形式で（機械語を使って）プログラムされるかについて基本的な理解ができるようにした。計算機のプログラムの作り方をいくつか例を示すのに単純な機械を使い，また計算機が実際動作する深さの程度や，プログラムのそれぞれの段階での簡単さなどを示してある。反復操作をするため，プログラムを計算機がどのように修正できるかも説明してある。

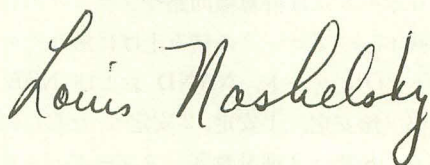
1章と2章で計算機の一般的な面を考察した上で，次の3章で論理的，数学的な点から計算機の回路の働きを述べる際に使われる数体系，コード，ブール代数について扱っている。

3章と4章は計算機の仕事で用いる数体系（2進数，8進数など）を扱い，印刷装置，紙テープ，磁気テープで，英字や数字を表わすのに使われる2進化コードなどについて述べる。これらのコードを使って，いろいろな英字や記号が計算機やそれについている入出力装置の中で処理される。

5章は，ブール代数，すなわち論理代数の入門である。論理的演算を数学的に述べたり，論理的な表現を簡単にする技法を考察する。その上，簡単化する

ために，カルノー図を利用した図解法も含まれている。論理的な技法を用いた例として，半加算器，比較器，全加算器，全減算器などがある。

6章～8章は計算機回路やブロックの詳細に及んでいる。既に前章で計算機の基本論理ブロックの積み上げに触れたので，6章および7章では AND ゲート，OR ゲート，NAND および NOR ゲートの回路，マルチバイブレータ回路（無安定，1安定，2安定），ならびにシュミットトリガ回路などを扱っている。8章は2進計数器，フィードバック計数器，シフトレジスタを作るために，上記の回路がどのように接続されるかを示している。計数器とシフトレジスタは計算機やディジタルシステムのいろいろな場所で使われている。それを使った例もいくつかあとの章で触れている。

この本の最後の部分は，計算機の別の部分について細かく触れている。9章で制御装置を扱い，全般的な複雑な操作をさせるのに，計算機にたくさんの簡単な段階を遂行させる基本信号を与えるため，どのように刻時信号が発生され，利用されるかを示す。単純な算術演算装置については10章で述べている。正負符号を含む加算器と減算器回路の動作や，乗除回路の動作などがそれである。

次に，11章では2つの記憶装置，磁気コアと磁気ドラムを詳細に触れている。情報がどのように蓄積され，装置全般がどのように組み立てられているかも説明してある。12章は汎用および特殊用の計算機に関連ある種々の入出力装置を扱っている。カード読取り器，紙テープ，磁気テープ，磁気ディスク，高速印刷機のような入出力装置が論じられている。特殊用の計算機に使われるアナログ-ディジタル，ディジタル-アナログ変換器の働きも含まれている。

この本にあげられた題目は，汎用の計数型計算機の領域ばかりでなく，ディジタル回路の一般領域においても重要である。計算機のような回路は，電力会社，電話会社，計測制御やマイクロウェーブ関係などの会社で使われている。技術部門のほんの一部の学生だけが，計数型計算機の仕事に専念するであろうが，全部とはいかないまでもほとんどの人々が計数型計算機やディジタル回路となんらかの関係を持って働くことになろう。ディジタル回路やディジタル装置の構成を基本的に理解すれば，もっと複雑な装置を理解するまでになるのである。

ニューヨークで浦さんと会って，私は計数型計算機の分野で，日本で行なわれている研究にかなり強い印象を受けた。日本の他の多くの大学のうちでも慶応大学は，私が合衆国でよく知っている大学と同じくらいに計数型計算機のハードウェアやソフトウェアに熱心に取り組んでいるように思える。浦さんが労

力と時間を費やして私の著書を翻訳してくれたことを喜ぶものである。

1968 年 6 月

Louis Nashelsky

Louis Nashelsky

序

この本を書くのを思いついたのは，私が Queensborough Community College ではじめて計数型計算機のコースを教えたときであった。その頃，最新の知識，十分かつ明確な実例，多数の問題などを備えた良い教科書はまだ手に入らなかった。ここで取り入れている題材は，大学で書いたり，用いたりしたものをもとにして，題材を教えやすくする必要から，それに数多くの改善を加えたものである。そのためこの本は，特に授業に向いたものになっている。

本文は3編——基礎，計算機回路，計算機装置——に分けられる。基礎編は基本的な数体系，計算機コード，ブール代数，機械語プログラミングに及んでいる。計算機回路編は数多くの，現在の計算機の論理回路や構成ブロックについて詳細に論じている。含まれている論理回路は AND, OR, NAND, NOR ゲートである。計算機ブロックは基本マルチバイブレータ回路——2安定（フリップフロップ），1安定（ワンショット），無安定（クロック）——やシュミットトリガ回路を含んでいる。計算機装置に関する編は記憶装置，制御装置，算術演算装置，入出力装置を扱っている。

この本には，学生にその章の重要な面を強調するようなたくさんの実例が入っている。また，問題を各章の中や終りにつけてある。問題が小分けして提示されているから，教員は扱っている題材にふさわしい問題を与えることができるし，完全にその章が終った後に，全部の題材に関係した問題を出すこともできる（問題を選んで，その解答をこの本の最後に与えてある。解答がついている問題には番号の下に線を引いてある）。この本は，また学生の計数型計算機の理解を助け，教員にはそれを提示する助けになるような写真や図がたくさんのせてある。

カリキュラムによっては，このうち特別な部分が他よりももっと重要であったり，既にいくつかの題材をそれ以前のコースで学んでいることもあろうが，その場合に対しても十分な題材が3編に収められている。たとえば，あるカリキュラムにおいて，計算機装置に関する講義があるなら，別のコースで計算機基礎および回路に関して，3章～8章を通じての題材を用いることができる。高等学校のプログラムには，基礎を扱っている最初の編だけしか使えないかもしれないが，たくさんの実例や問題はうまく利用できよう。また，別のカリキ

ュラムでは,コースの主要部分の基礎として,計算機回路やブロック(第II部)に関する題材だけを用いることもある。

本質において,この本で扱っている題材の量やその並べ方は短期大学における計算機コースに適当なものであるが,他の学校のいろいろのプログラムにおいても用いることができる。この本は編,さらに章に分けられていて,教員が講義の区切りをつけるのに都合よくなっている。

この本のために,実例となる題材を送ってくれた多くの会社に感謝の意を表したい。I. B. M. はたくさんの写真や図を都合してくれたが,それは数章(主に 12 章)で用いてある。他にもたくさんの写真を Burroughs Company, Digital Electronics Corporation, General Electric Corporation, Minneapolis-Honeywell Company などから提供された。

この本を書くにあたって, Queensborough Community College の電気技術学科の主任 Joseph B. Aidala 教授,同学科 Leon Katz 教授の援助と激励に謝意を表するものである。なお,原稿をタイプしてくれた Mrs. Sylvia Neiman, Queensborough Community College で用いた原本の制作を手伝ってくれた Mr. Nathan Blumkin, 出版にあたって最終原稿をタイプしてくれた Mrs. Ellen Zawel にそれぞれ厚くお礼を述べたい。

1966 年 1 月

Louis Nashelsky

目　次

1　入　門

1-1　概　説 · 1
1-2　計数型計算機システム · 2

第Ⅰ部　基礎編

2　機械語によるプログラミング

2-1　概　説 · 11
2-2　機械語によるプログラミング · · · · · · · · · · · · · · · 12
要　約 · 27

3　数体系

3-1　概　説 · 29
3-2　2進数体系（2進法） · 30
3-3　8進法およびその他の数体系 · · · · · · · · · · · · · · · 37
3-4　2進法の算術 · 42
要　約 · 49

4　コード

4-1　2進化10進数 · 52
4-2　3あまりコード · 54
4-3　2-5進コード · 55
4-4　コードのパリティ · 57
4-5　グレイコード · 58
4-6　ASCII 入出力コード · 62
要　約 · 64

5　ブール代数

5-1　ブール代数の基礎 · 66
5-2　リレー論理 · 74

5-3　電子式論理ゲート……………………76
5-4　カルノー図……………………84
5-5　論理技法の応用……………………90
要　約……………………99

第 II 部　計算機回路

6　計算機の論理ゲート

6-1　ダイオードによる AND ゲートと OR ゲート……105
6-2　トランジスタインバータゲート……………………115
6-3　トランジスタによる NAND ゲートと NOR ゲート……120
要　約……………………128

7　電子計算機の回路

7-1　2安定マルチバイブレータ回路……………………131
7-2　1安定マルチバイブレータ回路……………………137
7-3　無安定マルチバイブレータ回路……………………143
7-4　シュミットトリガ……………………147
要　約……………………151

8　計数器とシフトレジスタ

8-1　2進計数器……………………154
8-2　シフトレジスタ……………………158
8-3　フィードバック計数器……………………161
要　約……………………168

第 III 部　計算機装置

9　電子計算機のタイミングと制御

9-1　刻時信号……………………172
9-2　符号化および解読用マトリックス……………………175
9-3　制　御……………………179
要　約……………………182

10　電子計算機の算術演算

10-1　算術演算一般 · · · · · · · · · · · · · · · · · · · 185
10-2　算術演算装置，加算／減算 · · · · · · · · · · · · · · 188
10-3　乗　算 · 195
10-4　除　算 · 198
要　約 · 200

11　計算機の記憶装置

11-1　記憶装置概説 · · · · · · · · · · · · · · · · · · · 202
11-2　磁気コア記憶装置 · · · · · · · · · · · · · · · · · 204
11-3　磁気ドラム記憶装置 · · · · · · · · · · · · · · · · 215
11-4　その他の記憶装置 · · · · · · · · · · · · · · · · · 225
要　約 · 227

12　入出力装置

12-1　入出力技術一般 · · · · · · · · · · · · · · · · · · 229
12-2　穿孔カード · 230
12-3　穿孔紙テープ · · · · · · · · · · · · · · · · · · · 236
12-4　磁気テープ · 240
12-5　磁気ディスク · · · · · · · · · · · · · · · · · · · 247
12-6　高速印刷機 · 247
12-7　アナログ-ディジタル (A/D) 変換 · · · · · · · · · · 252
12-8　ディジタル-アナログ (D/A) 変換 · · · · · · · · · · 264
要　約 · 266

問題解答 · 269

索　引 · 272

0001＝1

入　門

1-1　概　説

計数型電子計算機（digital computer）は研究の重要な一分野であり，今後その利用が高まり，重要性がますます大きくなっていくことはまちがいない。そこで，計算機がどのように使われ，どのような働きをしているかについて知ることが必要になってくる。計算機の歴史は 19 世紀にさかのぼるが，現在のような計算機の思想がはじめて考えられたのは，1930 年代であり，実際に発達したのはつい 1950 年頃のことなのである。実に，現在の電子式計数型計算機（ディジタル コンピュータ）の歴史はほんの 15 年にしかならない。この短い間に計算機がかように進歩し，かつ科学的および事務的演算における重要な道具となろうとは信じられなかった。技術分野の発達と高速のデータ処理装置および計算機に対する要求とが相まって，この非常な短期間の発展を促したことは疑いないことである。

「コンピュータ」という言葉は一般用語であるから，まず計算機の種類について述べ，この本で取り扱う計算機がどの型に属するかを明らかにしなければならない。広い意味では，計算機には計数型計算機と**相似型計算機**（analog computer）とがあるが，この 2 つは操作上まったく異なったものである。計数型計算機は，いわゆる 2 進数の ONE と ZERO の世界で働き，驚異的な速度でその数字を処理するものである。現在のたいていの計算機では，40 桁の数の加算を 1 マイクロ秒（1 マイクロ秒は 100 万分の 1 秒）以下で処理できるほどになっている。言いかえると，1 秒あたり 100 万回も加算ができることになる。このように高速で算術演算を行なえるので，大量のデータを短時間で処理することができる。相似型計算機は，これに反し，現実の世界で動作するものであって，考える物理的な問題を電気的な量や機械的な位置で表わして処理する。

相似型計算機では，汎用のものでも，1秒間に数百サイクル以上の速さでは解を出せない。しかし速度が遅いからといって必ずしもその相似型計算機が貧弱な装置であるということにはならない。もしそうならば，計数型計算機が急速にそれにとってかわっていたであろう。実際，それぞれが特定の適用をうけてはじめてそれぞれのすぐれた能力を発揮するのである。事実，必要な操作を行なうために，この2種類のものを組み合わせて用いる分野がふえている。**ハイブリッド計算機**（hybrid computer，以上の結果生まれた，機能的な計算機を表わす言葉）は，航空とか航海のような分野でのある種の問題を解くのにますます重要になっている。現在，たいていの特殊用計数型計算機では，アナログ量の入力データを2進データに変換し，計算を行なった後，逆にそのデータをアナログ量に再変換する必要がある。相似型と計数型のはっきりした区別は，そのうちの少なくとも1つについて，もっとよく理解してはじめて生まれてくるものである。

計数型計算機はさらにまた汎用と特殊用の2つに分けられる。汎用のものはプログラムによって技術的な多くの問題を解けるように設計されている。数分の間に医学的な問題をかたづけてしまったり，財務簿記をつけたり，工業デザインの研究をしたり，人間を相手にチェッカをしたりする。特殊用のものは特別な問題むきに設計されていて，その種の問題だけを最もうまく処理するようになっている。したがって，それ相応にその特殊目的に対して小さく，安く，高速で問題を処理できる。この2つの例として，精錬所の生産制御やミサイルあるいは飛行機の誘導制御をあげることができる。計数型計算機の2つの型は構造上基本的にはまったく同じものである。この2つの持つ相異点は，データを読み込む装置や情報をとり出す装置の特徴と特殊用のものに比べて汎用のものは融通性に富んだ動作ができることにある。

1-2　計数型計算機システム

計数型計算機の基本構成は，入力装置，演算装置，制御装置，記憶装置，出力装置から成る。図 1-1 のブロック図は多くの計算機での流れを簡単に示したものである。まず，汎用の機械から考えてみることにしよう。入力装置には紙テープ，穿孔カード，磁気テープ，タイプライタ，磁気ディスクなどが多く用いられている。入力装置はデータや命令を計算機に送り込む働きをする。解く問題の種類をかえるには，これらの入力装置から，新しく命令やデータを

入れさえすればよい。各入力装置は，それぞれの用途を持っている。穿孔カードには個々のデータや命令を入れることができるが，これらは容易に入れかえたり，挿入したりできる。各カードは，ある特定の物や人を表わすのに使うことができる。たとえば，各学生の履修カード，各契約者の電話代や電気代の請求書，ある会社からの購入品，会社の在庫品などである。穿孔カードはこのような目的には，最も有用な入力手段の1つである。しかし，大量の情報を扱うときは，穿孔カードでは遅すぎる。このときには，磁気テープがよい。一度磁気テープに入れておきさえすれば，機械への入力はより速くでき，またデータを更新して同じテープに書き戻すこともできる。たとえば銀行では記録をテープに入れている。

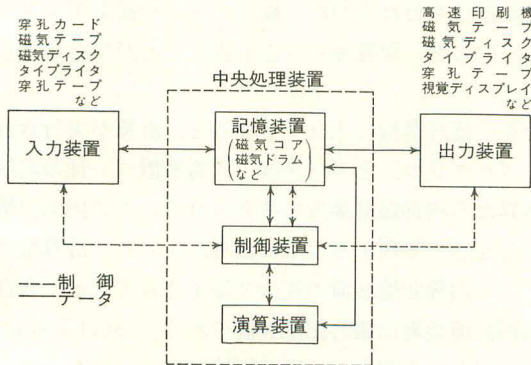



図 1-1 計算機の基本的なブロック図

営業上大量のデータを扱う保険会社でもまたそうである。契約者の勘定をするにあたって，磁気テープからデータを読み出し，機械の中で適当な処理を行ない，またテープにしまっておいたりする。大量のデータを処理することを考えると，磁気テープは操作上欠くことのできないものである。磁気ディスクのファイルは磁気テープとその用途が非常に似ているので，ここでは同じに考えることにする。記憶装置の章でそれらの詳細に触れることにしよう。タイプライタからの入力を使うことにより，計算機にどんな動作をするかを連絡したり，各装置の状態を尋ねたりすることが容易にできる。大量のデータを入れることはできないが，いわゆる計算機との会話（quick talk）ができるわけである。タイプライタは，また，計算機との連絡にほとんど時間がかからないような小さなプログラムにとって便利なものである（デバッキングや故障探索時の処理など）。穿孔テープは読み込みが速く，また穿孔カードに比べるとデータの保管

に装置や場所を必要としない。磁気テープには何百万ビットものデータを入れることができるが，穿孔テープは問題によって特定の長さに切ったり，つないだりするのが容易であり，切ったテープの処理も楽である。磁気テープにはいろいろな問題をたくさん入れておくことができるが，消してしまわないように注意して扱わなければならない。紙テープは数インチまたは数フィートの長さで使うのが普通であるが，永久記録である。しかし紙テープは磁気テープのように更新することができないという点で不便であると言えよう。

結論として，問題の多様性に応じて，また処理されるデータ量に応じてたくさんの違った入力装置が必要になることは明白である。それぞれ特別な作業領域に応じて最適のものなのである。実際，大きな計算機では，もっと融通性があり，より有効な操作を行なうのに，磁気テープや磁気ディスク，タイプライタなどはもちろんのこと，穿孔カードをも扱うことができるようになっているのが普通である。

データや命令を一度計算機に入れてしまうと，計算が実行される。一般に，命令（すなわちプログラム）とデータは，入力装置（一種の記憶装置である）とは動作速度が異なる内部記憶装置に蓄えられる。この内部記憶装置は，少量のデータを非常な高速で処理するように作られている。計算機の基本速度はたいていの場合，この内部記憶装置の速度で限定されている。現在使われている最も一般的な内部記憶装置は磁心記憶装置である。これは1秒あたり百万語(命令またはデータ）ぐらいの割合で処理が可能である。すなわち，1秒あたり百万回ぐらいの速度で，計算に必要な語を提供したり答を受け入れたりすることができる。しかし，記憶容量はつい最近まではほんの数千語ぐらいに限られていた。いまでは，数十万語の容量のものも設計されている。計算機は新しいデータを入れ，そのデータを更新し，そしてそのデータを読み出して，そこに他の情報を入れるが，それにつれて記憶内容は絶えず変化していく。

記憶動作は「制御装置」と呼ばれる中心的な装置で制御される。これは計算機全体も制御するいわゆる心臓部である。制御装置は記憶装置の中の命令を解釈し，どこからデータをとり，どこに送り，どんな操作を行なうかなどを他の装置に伝える。演算装置は，加算，減算，乗算，除算の操作を高速で行なうところである。1つの加算を終るのに多くの段階が必要であるが，それでも演算装置は記憶装置よりも高速で動作することができる。したがって，システムとしての動作速度が増大することになる。演算結果は再び記憶装置に戻される。情報は後に記憶装置から出力装置に読み出されることになる。それには，磁気テ

ープ，穿孔カード，タイプライタ，穿孔テープ，磁気ディスクまたは高速印刷機などがある（ほとんどの出力装置が入力機器としても使われる)。高速印刷機は，いま述べたもののうちで一番速い恒久的な目に見える記録で，大量のデータを扱うときに必要なものである。それぞれの詳細は 12 章に収めてある。

汎用計数型計算機を多目的でしかも役立つ装置にする最大の特徴は，**プログラム** (program) で制御できるという点にある。プログラムとは，計算機にどんな操作を，どんな順序で，どんなデータについてやればよいかを指示する記述（命令）のリストである。最近のプログラム用の言語では，簡素化された様式で命令を書くのが認められていて，計算機がしなくてはならない各段階をプログラムが細かく記述する必要がないようになっている。ほんの数命令で計算機に対する数百の動作を指示できるのである。大量の情報に対して計算を反復する機能が計算機に備わっていないと，まったく役に立たないものになるであろうということは記憶していなくてはならない。いわゆる計算する機械（カルキュレータ，calculator）はたくさんの有効数字まで，一度だけ直ちに解くのに使われる。同じ問題を解くのに，カルキュレータでは用いる数を何千回もかえることにはとても応じられないが，コンピュータ（この書物での計算機）にはやさしい仕事である。

プログラム用の言語の他に，計算機を改良してきたもう 1 つの重要な技術分野に，いわゆる**固体素子** (solid-state component) の使用があげられる。初期の計算機はリレー（継電器）を用いており，次に真空管に移った。リレーは遅すぎるし，かなりの電力を要するうえに場所をとり，十分に信頼できるものではない。真空管もまた寿命が短く，かなりの電力を要し，場所をとる。もし現代の計算機が何百万の部品を要し，1 つでも悪ければ誤動作を起こすということを考えれば信頼性の高い素子が必要なことは明白であろう。

初期の真空管式の計算機では数千の真空管を使用していた。テレビを持っている人ならだれでも真空管の破損によって，しばしば装置が使えなくなってしまうことがあるという事実を認めるであろう。もし真空管全部をある一定期間の後とりかえなければ，真空管が次々に悪くなり，その計算機は絶えず誤動作を起こすことになるであろう。固体素子たとえばトランジスタ，トンネルダイオード，マイクロ ロジック の回路などを使った現在の計算機では，これらの素子の故障による誤動作はきわめて起こりにくくなってきている。現在では，問題が起こることは少ないが，問題の大半は補助的な機械部分——カード穿孔機，カード読取り機，磁気テープ装置など——によるものである。固体素子使用によ

って，計算機の設計はより複雑になって，計算機の有効な動作時間の犠牲なしに記憶容量，制御能力，多様性をより多く持たせられるようになった。IBM のシステム/360 では，その改良された信頼性と マイクロ ロジック 回路の小さいことにより，非常に多くの働きを持たせている。また記憶容量を増すことができるようになって，計算能力はより増大した。図 1-2 には，大きな計算機システム（IBM システム/360 Model 40）と小型の計算機（DIGIAC 3080）の両方を示してある。

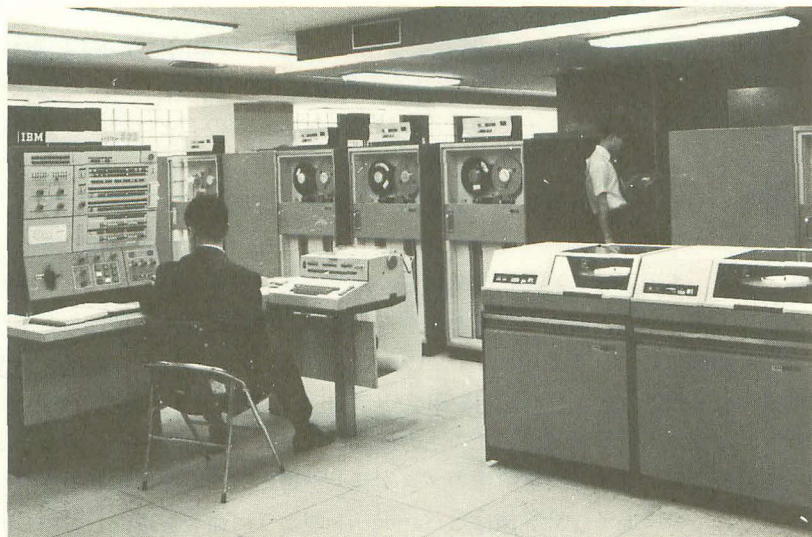
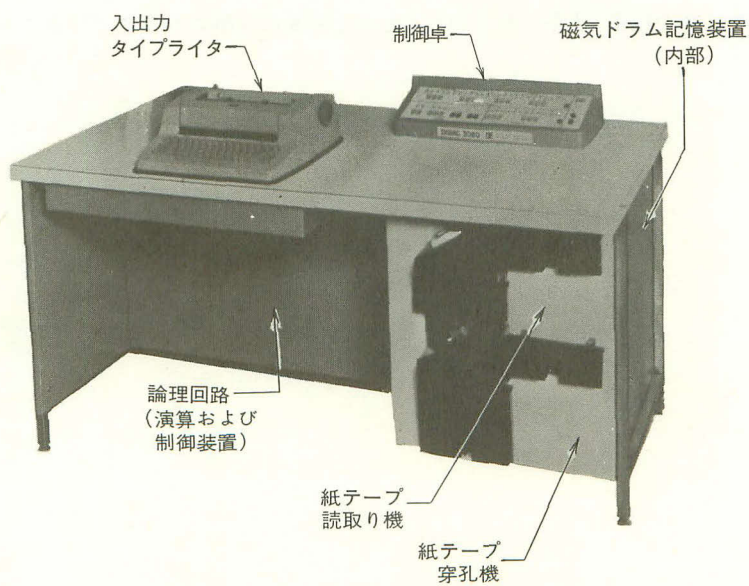
計数型計算機はある一群の演算を果たすように作られている。これには，2 つの数の加算，減算，乗算，除算がある。大容量の高速の電子記憶装置があって，その中に，演算に用いる数（データ），演算結果，および命令が格納される。命令が格納されるということはとりわけ重要なことである。さらに，計算機はこの格納されている数をとり出して，2 つのものが等しいかどうか「調べる」ため比較したり，正であるかどうか「調べる」ため符号を判断したり，記憶装置の中でデータの転送を行なったりできる。そして最後に新しいデータや命令を読み込んだり，記憶装置から計算結果を外にとり出したりすることができる。計算機の 1 つ 1 つの段階または動作は，根本的には非常に簡単なものであるが，これらの操作を高速度で行なうために，これらの動作の単純さが質的に変化するのである。この単純な動作をさらに高いレベルのものにまで高める第 2 の面は動作の正確さである。その正確さはそれぞれの数に使われる数字の桁数によって制約される。ある計算機では，扱われる数の長さ（2 進数字の桁数）が 64 ビット（2 進数）に定まっている。この場合，精度は相対的にみて 2^{-64} であって，気にするには及ばない。もちろん，非常に多くの計算を繰り返し行ない，その間に四捨五入など行なって，計算の回数が多くなるにつれて精度は落ちて，やっと満足する程度か，または悪くすると不十分になることがある。しかし，1 語あたりのビット数を大きくすることによってこれは改善される。このようにして，精度は望むまま常によく保持できる。相似型計算機ではこんなに精度を増すことはできない。計数型計算機にその単純な根本的動作とまったく異なった次元を与える最後の最も重要な面はプログラムである。計算機は，そのままでは基本的な動作として微分方程式を解いたり積分を行なうことはできない。だが，プログラムを組むことによって，2, 3 の簡単な動作だけを用いて，このような計算をすることができる。単に単純な演算に答を出すというのではなく，最終的な結果まで出す一連の演算を指定するのがプログラムである。そして，これらの演算を行なう速度は非常に速いので，大きな複雑な問題を解

(a) IBMシステム/360 Model 40

(b) DIGIAC 3080

図 1-2　典型的な汎用計数型計算機

くことが可能になる。

計算機が処理する分野は広がり,日ごとにその働きは印象的になっているが,そのすべてについて,実際の動作はなにも不思議なものではないし,また機械そのものが考えたり行動したりしているわけでもない。このことはいくら強調してもしすぎることはない。技術はまだそこまでには至っていない。このことは読者を驚かせるかもしれないが，計算機は本質的には愚かなものなのである。プログラムの形で与えられたそれぞれの命令を受けて，その演算を行なうだけなのである。もし，その結果，永久に答の出てこない終りのないループに機械を入れてしまったら，そのことがわからないまま，無限に動作を続けていくだろう。計算機は特に，その中にプログラムされたもののほかには，理性とか思考はまったく使わない。しかし，機械に入れられた情報は大きく成長するので,時おり，あたかも計算機自身でその出力に現われたものを作り出すかのように働いているように見える。計算機のプログラムを書く人が実際は思考や決定を行なっているのである。計算機はただロボットのように命令を遂行しているにすぎない。機械はただ述べられた簡単な動作を行なうだけということを心にとめておけば，計算機の動作は決して神秘的なものとは思えないだろうし，プログラミングの技術（2章で触れる）ももっと容易に理解されることであろう。

第I部　基礎編

0010 ── 2

機械語によるプログラミング

2-1 概　説

計算機についての一般的な事柄を1章で述べてきたので，これから，計算機を使って，どのようにして問題を解いていくかを考えよう。計算機に命令を下す基本的な“言葉”は**機械語**（machine language）と呼ばれる。機械はコード化された数字を使って命令されるが，その数字は実行すべき演算の種類とその演算のための数値をどこからとるか，またはどこに入れるかを示している。機械は**2進数**（binary digit）によって動作するので，実際に使われる命令は2進表示である。プログラマやオペレータの便宜のために，計算機の外では10進または8進数が使われることがあるが，それらの数は機械の内では2進数（または2進にコード化された数）に変換される。計算機の制御装置は機械命令を理解し，その動作の手順を進めるために必要な信号を発する基本的な役割を持っている。制御装置は大量の演算のために記憶装置と演算装置を使う。

問題を解くためにどのように計算機が使われるのかについて，いくつかのプログラム例を示して，その重要な特徴を明らかにしよう。機械語を考える主要な理由は，計算機の動作をその基本的なレベルで議論できるからである。このことは，あとで異なった計算機の装置を考えるときの助けになるであろう。プログラマは，どのように計算機が動作するかより，問題をいかにして解くかに興味を持っているのであって，機械語はまれにしか使わない。より高いレベルの計算機用言語の持つ価値については，機械語によるコーディングを説明したあとで触れることにする。

2-2　機械語によるプログラミング

機械語によるプログラミングを説明するのに，1次式 $Y=2X+6$ の値を求めるための簡単なプログラム（プログラム 2-1）を考える。

プログラム　2-1

記憶番地	命令 演算	命令 番地	説明
000	10	010	リセット(空にする)して2を加える。
001	20	012	2 に X を掛ける。
002	14	011	2X に 6 を加える（=Y)。
003	30	013	Y を格納する。
004	50	013	Y を印刷する。
005	00		停止する。
010	(2)		値 2 を格納。
011	(6)		値 6 を格納。
012	(X)		値 X を格納。
013	(Y)		計算された値 Y を格納。

左側にある番地は記憶装置のどの特定の場所を考えているかを示す。記憶装置の中の各語には番地が割りふられていて，その語を探すために使われる。これは郵便配達人が手紙の住所でどこに配達するかがわかるのとまったく同じである。記憶装置の中の特定の番地にはただ1つの語しかない。プログラムは順次実行すべき演算の順序を示す。このプログラムは記憶装置に格納され，そして計算機はその最初の番地の演算を行なう。命令は，なすべきことを機械に示す演算命令コードと番地(オペランド)から成っている。**オペランド** (operand) は演算を行なうためにどこから情報を得るかを機械に示すものである。演算が終了すると結果はアキュムレータに残ることを知っておく必要がある。**アキュムレータ** (accumulator) は計算機の演算装置の一部で，演算が終るとそこに算術演算の結果が残る。別の演算を行なうためには，アキュムレータの内容を，記憶装置の中に格納し，新しいデータをアキュムレータの中に入れなければならない。そのため，プログラムの中には，アキュムレータにデータを入れたり，次の演算のためにデータをアキュムレータからとり出したりする操作がたくさん入るのが普通である。演算装置は非常に速く演算を行なう。そして，そこで

使う情報は記憶装置からとり出され，演算結果は記憶装置に送り返される。入力データはまず記憶装置に入れられ，必要に応じて演算装置に移される。演算した答は，一時，記憶装置に格納され，必要なとき読み出される。このようにすれば，より遅い入力装置の速度で演算装置の演算速度がおさえられてしまうことはない。

ここの例では，010 番地から 013 番地までは，このプログラムのためのデータを格納するために使われている。データは記憶装置のどこに格納してもよい。というのは，一般に記憶内容をとり出すのに必要な時間は番地にかかわりなくほぼ一定だからである。プログラムは順次連続して格納されるが，計算機はプログラムの連続していない番地に分岐するようにプログラムにより指示されることもある。分岐命令については，後の例題で詳しく述べる。なお，記憶装置の中のデータは演算装置に読み出されても記憶装置に残っている（非破壊読出し）。もし新しいデータが記憶装置のある場所に格納されると，その番地に格納されていた古いデータは失われる。これはテープレコーダに録音するのと同じである。新しく録音すると，古い録音は消され，新しいものだけが残る。

000 番地から実行を開始すると，プログラムは，まず，加算装置（アキュムレータ）を零にして，それから 010 番地に格納されている数を加えるように命令する。アキュムレータに格納される数は 010 ではないことを，はっきり理解しておかなくてはならない。アキュムレータに入るのは，記憶装置の 010 番地に現在格納されている数である。このようにして，2（プログラムが始まる前に格納されていた値）が 0 に加えられ，アキュムレータには 2 ができる。次の命令では，012 番地に前もって置かれた数（この場合 X）はアキュムレータの中の数（ここでは 2）と掛け合わされる。その結果，積 2X がアキュムレータにでき上る。ここで 2X というのは，代数的表現ではなくて，X で表わされている数と 2 という数との積（具体的な数値）のことである。計算機は数の演算だけを行ない，結果も数であることを忘れてはいけない*。 X や Y という表現は数につけたレーベルである。第 3 の命令によって，011 番地に格納されている定数 6 を 2X に加え，その和 2X+6 がアキュムレータに残る。これは Y そのものであって，013 番地に格納される。その次の命令では 013

*〔訳注〕 数の演算も，もとをただせば，後の章で述べるように AND や OR の論理演算の組合せで実行される。計算機によっては，論理演算の命令コードも備えている。単純な算術演算や論理演算などを組み合わせることによって，計算機は，単に計算する機械にとどまることなく，翻訳とか作曲などを含めた広義の情報処理に使われる。

番地の内容すなわち Y が印刷される。009 番地にある最後の命令は，計算機にプログラムが終ったことを教え，このプログラムについての動作は終了する。

上にあげた例題で，演算の種類は命令語の左部分にある 2 桁の数（10, 14, 20 など）で示されている。この書物の例題では，表 2-1 にあげてある命令とコードを使うことにする。

表 2-1　機械語コード

コード	演　算	説　明
00	停止	機械にプログラムの終了を知らせる。
10	リセットし加算	アキュムレータを零にし，そこにオペランドの番地* に格納されている数を加える。
14	加算	オペランドの番地にある数を現在アキュムレータにある数に加え込む。
15	減算	オペランドの番地にある数を現在アキュムレータにある数から引く。
20	乗算	オペランドの番地にある数にアキュムレータにある数を掛ける。結果はアキュムレータ。
24	除算	オペランドの番地にある数でアキュムレータにある数を割る。結果はアキュムレータ。
30	格納	アキュムレータにある数をオペランドの番地に格納する。
44	分岐	次の命令をオペランドの番地からとることを計算機に指示する。
45	負なら分岐	現在アキュムレータにある数が負なら次の命令をオペランドの番地からとる。そうでなければ，次にある命令に移る。
46	正なら分岐	現在アキュムレータにある数が正なら次の命令をオペランドの番地からとる。そうでなければ，次にある命令に移る。
47	零なら分岐	現在アキュムレータにある数が零なら次の命令をオペランドの番地からとる。そうでなければ，次にある命令に移る。
60	読込み	入力装置からオペランドの番地に新しいデータを読み込む。
50	印刷	オペランドの番地に格納されている語を印刷する。

*〔訳注〕 オペランドの番地 (operand address) とは命令のアドレス部に書いてある番地のことである。

2番目の例として，$Y=X^2+8X+12$（プログラム 2-2*）を求めるプログラムを考えよう。

プログラム 2-2

記憶番地	命令 演算	番地	説明
000	10	020	アキュムレータをはらってXを加える。
001	20	020	X に X を掛ける。
002	30	023	X^2 を格納する。
003	10	020	アキュムレータをはらってXを加える。
004	20	021	X に 8 を掛ける。
005	14	022	8X に 12 を加える。
006	14	023	8X+12 に X^2 を加える。
007	30	023	$Y=X^2+8X+12$ を格納する。
008	50	023	Y を印刷する。
009	00		停止。
020	(X)		X の値を格納。
021	(8)		値 8 を格納。
022	(12)		値 12 を格納。
023	作業用番地		中間結果を一時記憶するための格納場所。

020-023 番地は定数8と 12 ， X の値，計算の中間結果を格納するために使われている。まず最初に，計算機は 000 番地の命令を見る。そしてアキュムレータを零にし，020 番地に格納されている数をアキュムレータに加えるように指令する。次に，001 番地の命令を見て，020 番地にある値にアキュムレータにある値を掛ける。X が現在アキュムレータにあり，また，その X は 020 番地の値なので，この演算の結果アキュムレータに積 X^2 が残る。アキュムレータを別の演算に使いたいので，次のステップで中間結果の X^2 は 023 番地に格納する。それから，X の値を再びアキュムレータに加える。その際，残っている X^2 を消すためにアキュムレータをまず零にする。定数 8 が X と掛け合わされてアキュムレータには 8X ができる。005 番地にある次の命令は 12 をアキュムレータの内容に加えるよう計算機に指令する。その結果 8X+12 ができる。最後に，前もって計算されている X^2 を 8X+12 に加える。そして Y ができる。Y の計算された値は 023 番地に格納され（前にあった X^2 の値は消される），そして印刷される。最後の命令は，計算機に記憶装置

*〔訳注〕 プログラム 2-2 ではこの式のままの順序で計算しているが，$X(X+8)+12$ として計算した方が乗算の回数が少なくてすむ。

の中の命令をたどっていくのをやめるように指令する。020 番地まで進んで格納されている数 X を命令と思ってしまうことを防ぐのがこのステップである。計算機は非常に愚かで，格納されている値が数か命令か見分けることができない。プログラムは計算機を 1 ステップずつ進めていくが，格納された数を命令として読むようなことがあってはいけない。このプログラムでは，データが格納されている 020 番地までステップが進まないようにするために停止命令が使われている。

ここまでは，一連の演算で 1 つの値を求める単純なプログラムを考えてきた。多分，このような問題ならプログラムを書くより速く解くことができるだろうから，計算機に偉大な価値があることを示したことにはならない。計算機を使うということは，大量の解答を必要とする問題を解きたい場合に違いない。言いかえれば，計算機は，多くの異なった数値に対し同じ計算をして，数十万回でも答を出すことができる。そのような場合は，プログラムを書き，計算機にめんどうな計算をさせることは効果的であると言える。人間が計算すると一生かかるような問題も，計算機なら数分で計算することができる。

何回も繰り返して答を求めるプログラムの簡単な例（プログラム 2-3）を考えてみよう。計算の目的は 1 から 100 までの数の 2 乗を計算し印刷することで

プログラム　2-3

	命令		
記憶番地	演算	番地	説明
000	10	020	アキュムレータをリセットし，020 番地に格納されている数を加える。
001	14	021	アキュムレータに 1 を加える。
002	30	020	020 番地に格納する。
003	20	020	自分自身を掛ける。
004	30	023	2 乗した値を 023 番地に格納する。
005	50	023	2 乗した値を印刷する。
006	10	020	リセットして 020 番地の数を加える。
007	15	022	その数から 100 を引く。
008	45	000	アキュムレータが負なら 000 番地に分岐。
009	00		停止。
020	00000		値（最初は零）を格納。
021	00001		サイクルごとの増分 1。
022	00100		最終判定値 100。
023	作業用番地		一時記憶のための場所。

ある。卓上計算機を使ったとしてもこの問題はかなり時間がかかるが，計算機は数秒で計算してしまう。そしてプログラムも短い。

繰り返しの演算の鍵は“負なら分岐”の命令である。分岐または飛越命令はプログラムをたどる正常な順序をかえてしまい，記憶装置の新しい場所に移ってそこから次の命令をとってくるようにする。しかし分岐命令の条件に合致しないときは，その命令の次に移るだけである。機械語コードの表には，4 つの異なった分岐命令が示してある。最初は無条件分岐（44）であって，これは指示した番地から次の命令をとることを計算機に命じる。そこで，44 216 は常に次の命令を実行するために 216 番地に分岐することを機械に命じる。216 番地の命令が算術演算命令（はっきり言えば分岐命令でないもの）であると仮定すれば，機械はその命令を実行し，次の命令をその次の番地（この場合 217 番地）からとる。

3 つの条件つき分岐命令は，もう 1 つ別のオペレーションがあるということを除けば無条件分岐と同じような働きをする。計算機は，分岐命令を実行するか無視するか決定するために，アキュムレータの符号またはその値を調べる。たとえば，もし命令が 45 216 で，現在アキュムレータにある数が負なら，次は 216 番地に分岐する。2 つの方向すなわち 2 つの異なった行動が分岐命令により可能となる。しかしそのうち 1 つだけが実行される。この選択すなわち決定のオペレーションが計算機に“考える”機械の印象を与えることになるのである。プログラムの中の多くの分岐命令によって，計算機はいろいろな処理手順をたどることができる。それによって，種々の条件のもとで複雑な問題を解くことが可能となっている。例題のプログラムに戻り，そこで分岐命令がいかに使われているかを見てみよう。

020, 021, 022 番地の内容は，どんな値で計算を始めるか，その値は計算ごとにどれだけ増すか，どのような値で計算を終了するべきかを指示するものである。計算のはじめで計算機はアキュムレータを零にし，そこに 00000 を加え，それからその数に 00001 を加える。1 からすぐ始めることもできるが，020 番地の中の値を 021 番地の中の値だけ増すという形を，次の計算のサイクルの前に作りたかったわけである。この値はまたこのサイクルで 2 乗を作ろうとする数である。第 1 のサイクル（ループ）では，値は 00001（すなわち 1）で，この値にそれ自身を掛ければ，2 乗の値が得られる（002 および 003 番地の命令）。2 乗した値は 004 と 005 番地の命令により格納され印刷される。次のステップでは，次のループをさらに実行するか計算を終了するかを考える。アキュム

レータを零にして，そこに現在の値を加えた後，最終判定値を引く。もし最終判定値が現在の値より大きければ，少なくとももう1回ループを実行しなくてはならないので，負なら分岐の命令によって，次に000番地に戻り，そこの命令を実行する。この分岐命令は計算機に再びはじめから計算をするように命じているわけである。ループのはじめで，プログラムは2乗する値を1ずつ増すから，次に2乗する数は前回のループのときより1大きい。このループの繰り返しは，負なら分岐の命令においてアキュムレータの値が零（または正）になるまで続けられる。すなわち，計算機に十分なだけループをまわったことを教え，プログラムは次の停止を命じるステップに移る。

この同じプログラムは，020, 021, 022 番地に格納してある初期値，最終判定値および増分の値をかえることにより，任意の数の組について2乗を計算するのに使うことができる。たとえば，初期値 00025，増分 00005，最終判定値 02000 ならば，計算機は 25, 30, 35, 40, …, 2000 に対してその2乗の値を計算する。一度プログラムを書いてしまえば，いろいろな計算をするためには特定の値をかえるだけでよい。このことは計算機の持つ非常に強力な特徴の1つである。たとえば，保険会社で契約者の勘定についてのプログラムを一度作ってしまえば，同じ計算を何人の客について行なうかは簡単にかえることができ，短時間のうちに計算を終えることができる。

負なら分岐 の命令をループの問題にどのように使えるかについて理解できたかどうか，次の問題を試みてみるとよい。

問題 2-1　$Y=3X^2+2X+6$ について，X に 1 から 50 までの値を与えて，Y を求め，印刷するプログラムを書きなさい。

問題 2-2　1 から 61 までの奇数について，3 乗を計算し，印刷するプログラムを書きなさい。

問題 2-3　$Y=3X^3+2X$ の値を計算し，印刷するプログラムを書きなさい。ただし X は 250 以下の偶数全部。

3つの数の中の最大のものを見つけるプログラム 2-4 を作るのは比較的むずかしいが，判定の基本的過程をよく示している問題である。このプログラムは1組の3つの数についてだけ判定を行なうようになっているが，新しくいく組かの数を読み込んで演算を繰り返すように修正することは簡単である。

プログラム 2-4 を見ても，どんな方法で問題を解いているかはすぐにはわからない。プログラムを組む方法を論じる前に，まず問題をどう解析するかを考えてみよう。プログラムを書く前に，問題を正しく理解していなければなら

プログラム 2-4

格納場所	命令		
(番地)	演 算	番 地	説 明
100	10	120	リセットし X を加える。
101	15	121	X から Y を引く;(X−Y)。
102	45	108	もし Y>X なら 108 番地に分岐する。
103	10	120	X>Y; リセットし X を加える。
104	15	122	減算; X−Z。
105	45	113	もし Z>X なら 113 番地に分岐する。
106	50	120	X を最大数として印刷する。
107	44	114	114 番地に分岐する。
108	10	121	Y>X; リセットし Y を加える。
109	15	122	減算; Y−Z。
110	45	113	もし Z>Y なら113 番地に分岐する。
111	50	121	Y を最大数として印刷する。
112	44	114	114 番地に分岐する。
113	50	122	Z を最大数として印刷する。
114	00		停止。
120	(X)		
121	(Y)		
122	(Z)		

ないということはプログラミングのもつ主要な特徴の1つである。思考の過程を容易にするため，よく **フローチャート** (flowchart) を作る。 問題を解くのに必要なステップを，どのようにフローチャートで表わしているか見てみよう。図2-1では，第1のステップはXをアキュムレータに入れることである。次に X からYを引き，2つの方向のうち1つにプログラムを分岐させる。もしXがYより大ならば，XからZを引き，どちらが大きいかを調べる。ここにも2つの可能性がある。もしXがZより大なら，Xが最大であるから，Xを印刷して停止する。もしZの方が大きければ，Z が最大であるので，Z を印刷して停止する。X−Y の結果，XがYより小さいことがわかると，プログラムはY (Xより大) からZを引く。もしYがZより大ならYを印刷して停止する。もしZがYより大ならZを印刷して停止する。フローチャートは，以上説明したように，どのようにして問題を解くかを表わしている。問題を解く方法が一度決定されると，その演算を行なうためのプログラムを書くことができる。

機械語プログラムを見ると，第一の命令はアキュムレータを零にしてそこにXを加える。そしてXからYを引き，その結果の符号を用いてどこに分岐する

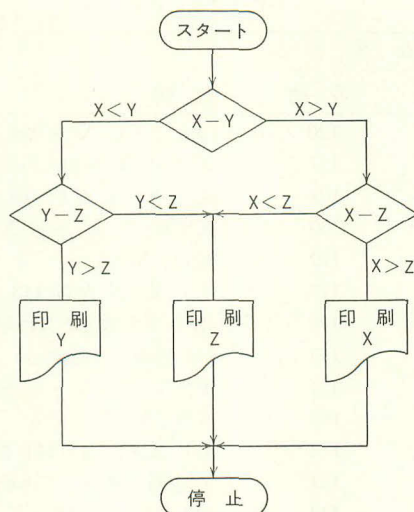



図 2-1　3 つの数の最大を見つけるためのフローチャート

か判断する。もしYがXより大（アキュムレータの符号は負）ならば，分岐命令により，プログラムは108番地から次の命令をとってくる。もし符号が正なら,次の命令はすぐ次からとられる。このことはXがYより大の場合に起こる。その次の命令（103 番地）はアキュムレータを零にしてXを加える。これはアキュムレータに前のステップでの差 X−Y が残っているためである。XとYのどちらが大きいかが決定されたので,次のステップではZとの比較を行なう。すなわち，次の命令（104 番地）ではXからZを引く。もしXがZより小さければ，プログラムは113番地に分岐する。もしXがZより大なら，次の命令で120 番地に格納されている数（この場合X）を印刷する。そして飛越（無条件分岐）命令によりプログラムは停止命令に進む。Xが最大である場合には，XからYを引き，そして再びXからZを引き，Xを印刷し，最後にオペレーションを終るために停止命令に進む。一般に,YやZが最大となる場合もあるから,最大値として Y または Z を印刷するオペレーションをプログラムに含めてある。YがXより大ならば，102番地の命令により，103番地から107番地までのステップは飛び越される。108 番地の命令によりYの値をアキュムレータに入れて，減算 Y－Z が実行される。そして再び分岐命令で判断が行なわれる。もしYがZより大なら，次の命令（111 番地）に進み，Yを最大の数として印刷する。そして次にステップ113を無条件に飛び越して，114番地の命令に進み，

計算機に停止を命じる。ZがYより大きかったなら，110番地の命令の結果，次の命令を113番地からとり，Zを最大数として印刷し，114番地の命令に進んで停止を命じる。1回だけ3つの数を比較するのだったら，このようにめんどうなことをしてプログラムを書くことはばかげている。どちらが大きいかは見ただけで判断することができる。しかし，1000組の3つの数について考えるなら，計算機を使うということは実際的であろう。そのときには，各組について上の手順を繰り返すため，いくつかの命令をプログラムに追加すればよい。それは，新しい X, Y, Z を読み込む命令とプログラムの最後から無条件にプログラムの最初に戻る分岐命令とを追加することである。計算機はデータがなくなったときに停止する。また，必要な命令を追加すれば，指定した回数だけループを回ったとき停止するようにすることもできる。なお，上のプログラムでは，3つの数のうち2つまたは3つの数が等しい場合については考えていない。演習として，これらの場合を考慮して，厳密にフローチャートを書き，このプログラムを解釈してみるのがよい。

いくつかの組について，上のオペレーションが行なえるようにすることを考

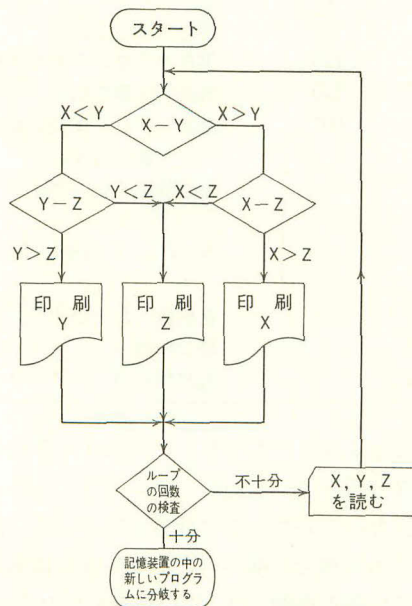



図 2-2 3つの数 X, Y, Z の最大を見つけるためのフローチャート

えよう。最大の数を見つけるという基本的な手順は同じだから，100 から 113 番地までは同じプログラムが使える。しかし，114 番地で停止するかわりに，X, Y, Z の値を新しく読み込んで，それぞれ 120, 121, 122 番地に入るようにしたい。114 番地には最大の数が計算（および印刷）されてから到達するから，そのプログラムを再び実行する前に，新しい一組の数を読み込むようにすればよい。十分な回数だけ計算を行なったかどうか判断するために検査が行なわれる。たとえば，来週は 2000 組の数があるかもしれないが，今日は 1000 組の処理をしなければならないものとしよう。機械によっては読み込みの際，新しい

プログラム　2-5

番地	命令 演算	 番地	説明
097	60	120	X, Y, Z の新しい値を読む。
098	60	121	
099	60	122	
100～113	プログラム 2-4 参照		
114	10	123	初期値をアキュムレータに入れる。
115	14	124	増分を加える。
116	30	123	現在のカウントとして格納する。
117	15	125	最終判定値を引く。
118	45	097	負なら新しく 1 組の数値を読み，このループを繰り返す。
119	44	—	記憶装置の別のプログラムに分岐する。
120	(X)		X, Y, Z の格納場所。
121	(Y)		
122	(Z)		
123	00000		初期値（およびカウント）。
124	00001		増分の値。
125	01000		最終判定値。

データがないならば停止するものもある。たとえば，穿孔カードの読取りの場合にはたいていそうなっている。しかし，このオペレーションを大きなプログラムの一部分として行なっていて，この計算が終ってから記憶装置の別の部分にある新しいプログラムに移って欲しいことがしばしばある。このようなときには検査が必要である。前の例題では，この検査を行なうために初期値，増分，最終判定値を記憶装置のどこかに格納しなければならなかった。この修正され

た問題のフローチャートを図2-2に示し，追加するステップをプログラム2-5に示す。

改良されたプログラムは097番地から始まる。まず，新しいX, Y, Zの値がそれぞれ120, 121, 122番地に蓄えられる。それからプログラムの主要部分が実行され，最大値が印刷される。114番地の命令が検査の始まりである。123番地に格納された初期値がアキュムレータに入れられる。この値を増加(この例では1だけ）して，それを123番地に格納する。アキュムレータには新しいデータが入れられなかったので,そのままの値が残っている。次の命令では，このアキュムレータの内容から最終判定値を引く。もし最終判定値の方が大きければ，プログラムは097番地に分岐して,そこから命令をとるように命じる。そして新しいX, Y, Zについてこのループを繰り返す。この手順はステップ118の検査でアキュムレータの値が零か正になるまで続けられる。この問題では，1000回の計算のあとで，そうなる。同じプログラムで新しい最終判定値をセットしカウントを入れた場所をリセットすれば，ごくわずかな手間だけで再び別の多くの数の組について計算することができる。すなわちプログラムを書きかえる必要はなく，単に，最初に新しい最終判定値を読み込めばよい。

プログラムが終了し，ステップ118にある負なら分岐が実行されないとすると，次の命令で別のプログラムの計算に移る。このように，計算機の動きを止める必要は決してない。新しいプログラムの一部で，あとで使うために，さらにプログラムを読み込んでいるかもしれない。事実，たいていの大型計算機は絶えず動いている。計算機は数分（あるいは数秒）の間に数学の問題を解い

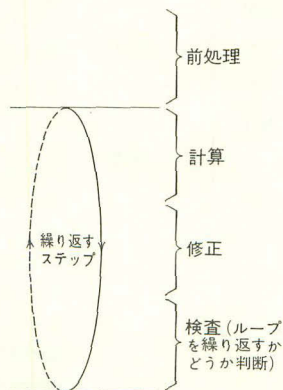



図 2-3 繰り返し計算のためのプログラムの型

たり，多くのデータの表を作ったり，学校のすべての生徒の成績を計算したりすることができる。それらのプログラムは例で示したようなステップがたくさん集まっただけのもので，計算機にとってはまったく同じである。機械はただ計算し続け，そして大量の情報を出力する。

計算の繰り返しで問題を解くことは計算機のプログラムで非常に重要なことであるから，その基本的なパターン（図 2-3）を調べてみよう。前処理のステップでは，カウントを零にし，適当な最終判定値と増分の値を与える。プログラムが終ったとき，カウントの値はそのときのまま残されている。同じ手順を再び使う前に，カウントを零にしなければならない。プログラムの準備を確実に

プログラム 2-6

番地	命令 演算	命令 番地	説明	
200	10	600	前処理	カウントを零にする。
201	30	300		
202	60	301	前処理	最終判定値を読む。
203	60	302		増分の値を読む。
204	60	250	計算	Xの新しい値を読む。
205	10	250		リセットし，Xの値を加える。
206	20	250		Xを掛ける。
207	30	251		X^2(＝Y) を格納する。
208	50	250		Xを印刷する。
209	50	251		Yを印刷する。
210	10	300	修正	現在のカウントをアキュムレータに入れる。
211	14	302		増分を加える。
212	30	300		新しいカウントとして格納する。
213	15	301	検査	
214	46			新しいプログラムへ分岐する（もし正なら）。
215	44	204		ループを繰り返す。
250	(X)			X の格納場所。
251	(Y)			Y の格納場所。
300				カウントの格納場所。
301				最終判定値。
302				増分の値。
600	00000			定数 0。

するためには，必ず前処理をすることにしておくとよい。それから計算を始める（方程式を解くとか比較をしたりする)。その計算が目的とする演算なのであるが，制御のために他のステップを実行する必要がある。修正のステップでは，カウントの値を決められた値だけ増加する。検査は必要な回数，計算が行なわれたかどうか調べるためのものである。もし必要なだけ計算したならば，新しいプログラムにいくか，あるいは一時停止するよう計算機に命じる。そうでな

プログラム 2-7

格納番地	命令 演算	命令 番地	説明	
000	10	115	前処理	カウントを零にする。
001	30	100		
002	10	100	計算	
003	20	100		
004	20	110		
005	30	101		$20X^2$ を格納する。
006	10	100		$7X$ を得る。
007	20	111		
008	14	112		6 を加える。
009	14	101		$20X^2$ を加える。
010	30	101		$Y=20X^2+7X+6$ を格納する。
011	50	100		X を印刷する。
012	50	101		Y を印刷する。
013	10	100	修正	
014	14	113		X を 1 だけ増す。
015	30	100		
016	15	114	検査	最終判定値を引く。
017	47	—		新しいプログラムに分岐する（もし零なら)。
018	44	002		ループを繰り返す。
100				X の格納場所。
101				Y の格納場所。
110	00020			定数 20。
111	00007			定数 7。
112	00006			定数 6。
113	00001			定数 1。
114	00011			最終判定値 (11)。
115	00000			定数 0。

ければ，検査のステップは計算機に少なくとももう1回ループを実行するために計算のはじめ（前処理の部分ではない）に分岐するように命じる。この手順を例題を使って説明してみよう。$Y=X^2$ の値を 100 個のXの値について計算する問題（プログラム 2-6）を考える。

前処理のステップではカウントを零にし，最終判定値と増分の値を読み込む。これらの値はプログラムが使われるたびに異なっていると考えられる。プログラムの計算の部分では，X^2 を計算し，そして X と $Y(=X^2)$ とを印刷する。修正のステップでは，増分の値を現在のカウントに加え，新しいカウントの値として格納する。検査のステップでは，もし新しいカウントの値が最終判定値より小さいならば，ループを繰り返させる。新しいカウントの値が最終判定値より大きくなると（アキュムレータが正になると），計算機は別の新しいプログラムに分岐する。

次の例題は，0から 10 までの X について，$Y=20X^2+7X+6$ の値を求める問題（プログラム 2-7）である。プログラムは簡単で容易にたどれるから，詳しい説明は省く。しかし，このプログラムには前回の例題とわずかに違う点があるので，この点だけを注意しておこう。

その1つは，増分と最終判定値は特別に読み込まれるのではなくて，他の定数と一緒に格納されていることである。このプログラムには，あまり融通性はなく，他の最終判定値や，増分の場合には使うことができない。前処理において新しい値を求めるかわりに，プログラムの中の定数の変更が必要になる。第2に，Xの値とカウントとは同じであるということである。そのため，Xを増加させると，それは新しいカウントとして，また次のループのXの値として使うことができる。第3に，最終判定値は 10 ではなくて，11 である。このプログラムでは，X は検査の前に1だけ大きくされるから，検査でXが 10 のとき，答は $X=9$ についてまで計算されている。$X=10$ までの答を出したければ，最終判定値は 11 にしなくてはならない。ループしたプログラムで問題を解く場合，以上2つの方法（計算の前にカウントを増す方法と，計算のあとにカウントを増す方法）が使われるが，一般にどちらを用いるかは要求される数学的計算の性質によって決まる。

問題 1-4　次の問題を解くプログラムを書きなさい。

1. 0 から 40 までの X の値について，$Y=X^3$ を計算し，各 X の値について X と Y とを印刷する。
2. 2 から 15 までの X の値について，$Y=3X+17$ を計算し，各 X の値について

X と Y とを印刷する。

3. Y の値を 3 として，1 から 15 までの X の値に対して，$Z=3Y^2-6X+12$ を計算し，各 X の値について X と Z を印刷する。
4. X の値を 3 とし，1 から 15 までの Y の値について，$Z=3Y^2-6X+12$ を計算し，そして各 Y の値について Y と Z を印刷する。
5. 10 組の X と Y について，$Z=3Y^2-6X+12$ を計算し，X, Y, Z を印刷する。各組のデータは計算をする前に読み込む。

機械語でのプログラミングを考えてきたが，長いプログラムを作るのは，確かにわずらわしいし，多くの数を扱うのでまちがいも起こりやすい。そのため1950年代の中頃，計算機を使う人達は日常語に近い形で命令を書ける問題向き言語を開発した。しかし，これらの言語（FORTRAN*, ALGOL** など）は，計算機で扱えるようにするため，機械語に変換しなくてはならない。この**ソースプログラム**（source program）を機械語の**オブジェクトプログラム**（object program）に翻訳することを**コンパイル**（compile）と呼ぶが，この作業も計算機を使って行なわれる。プログラムは機械語のかわりに，たとえばFORTRAN言語で書くことができ，それは計算機により機械語に翻訳される。実際に計算機を動かしているプログラムはすべて機械語であるが，このようにすれば，めんどうな機械語でプログラムを書かなくても機械語のプログラムを得ることができる。

要 約

この章では，計数型計算機のプログラミングに使われる基本的な言語について述べた。計算機に対する命令は数字であって，それは計算機によって2つの基本的な部分に分けて解釈される。すなわち命令コードとオペランドである。プログラムは順次解釈されて機械に指示を与えていくが，分岐命令によって，ステップをループさせて使うこともできる。多くの例をあげて，プログラムとはどのようなものか，そして問題を解くためにどのように使われるかを示した。

この章から導かれる本質的な事項は，計算機はただ単純な動作を行なうだけで，問題を解くのはプログラマであるということである。計算機は記憶装置の

*〔訳注〕 IBM社が開発した言語で，ISOで国際的な標準言語としてまとめられつつある（浦昭二編，FORTRAN 入門，培風館）。

**〔訳注〕 欧州の計算機関係者が提案し，後に米英の学者が加わって改善した国際的標準言語（森口繁一編，ALGOL 入門，日科技連出版社）。

中のプログラムに従って問題を解き，結果を紙に書いたり，他の手段で外に表示したり，テープやディスクに入れたりする。

問　題

1. 0 から 25 までの X に対して，$Z=8X^2-7X-12$ を計算するプログラムを書きなさい。

2. 10 から 45 までの Y に対して，$X=3Y^3+9Y^2-7Y+125$ を計算するプログラムを書きなさい。

3. 1 から 100 までの整数を順に印刷するプログラムを書きなさい。

4. 1 から 50 までの奇数 M に対して，$N=6M^3+3M+2$ を計算し，印刷するプログラムを書きなさい。

5. 100, 101, 102 番地に格納された 3 つの数のうち，最大のものを 200 番地に格納し，最小のものを 300 番地に格納しなさい。プログラムは 000 番地から入れるものとする。

6. 300 番地に格納してある数が偶数か奇数かを判断するプログラムを書きなさい。もし偶数なら 0 を印刷し，奇数なら 1 を印刷しなさい。

7. 100-105 番地に格納されている数の中に互いに等しいものがあるかどうかを調べるプログラムを書きなさい。

8. 200-210 番地に格納されているすべての数が等しいかどうか調べるプログラムを書きなさい。

9. 10 個の数を 100-109 番地に読み込み，もう 10 個の数を 250-259 番地に読み込むプログラムを書きなさい。

10. 100-109 番地に格納されている 10 個の数を印刷し，それから 250-259 番地に格納されている別の 10 個の数を印刷するプログラムを書きなさい。

0011 ＝ 3

数 体 系

3-1 概　説

いままで，10 進の数体系（数の表わし方）だけを学んできた人達にとって，別の数体系の導入は，驚きととまどいを感じさせるであろう。理解しやすいように，10 進数による表わし方を添えて例示しながら説明していくことにしよう。たいていの人は，10 進数体系は簡単だと思っているだろう。別の数体系での規則と演算を説明していくと，いままでなじんできた 10 進法の計算の中に，非常に基本的なアイデアが含まれていて，それを直観的にしか学び理解してこなかったが，いまや論理的に験証しなければならないことに気づくはずである。この数に関する基本的な演算と法則を理解できれば，別の数体系も同様に理解できるはずである。実は，電子計算機に関しては，これから述べる別の数体系の方が，理解し利用するのに，より簡単なのである。

10 進の数が与えられたとき，その実際の大きさを頭の中で作ってみたりなどはしない。たとえば 576 という数は 5 つの百と 7 つの十と 6 つの一とでできていることは明らかである。10 進法では，基本となる数字は，0 から 9 までであって，10 という数字はない。この場合の 0 という数字は非常に大切な数字である。10 という数は 10 進数体系での基数（base）と呼ばれ，1 個の十と 0 個の一である。数体系を考えるとき，数字の桁位置という概念を認めなければならない。数字の桁位置というのは，数字として零（0）を使うことから始まる非常に重要な概念であって，これによって，061 というのは 0 個の百と，6 個の十と，1 個の一の意味を持ち，610 というのは，6 個の百と 1 個の十と 0 個の一の意味を持つ。10 進法での数字の桁位置は数の大きさを決定するものである。この概念は簡単なようであるが，数理科学の基礎に輝かしい貢献をした昔のギリシャ人は，大きな数を表わすのに，どんどん新しい文字を導入していく

数体系を用いていた。現在の桁位置によって数の大きさを決める方法に比べて，この方法は，算術演算をする場合に，きわめて不便で損をしていた。

623 という数があるとする。これは，もちろん 6 個の百と 2 個の十と 3 個の一を意味する。これは，また，次のように書ける。

$$623=6\times10^2+2\times10^1+3\times10^0$$

ここで，10^2 は 10 の平方すなわち 100
10^1 は 10
10^0 は 1（定義により）

上の例で，基数 10 というものがどのように働いているかわかったと思う。数体系の基数というのは，これの 0 乗が一番右の位置の値となり，1 乗および 2 乗がそれぞれ右から 2 番目，3 番目の位置の値となるような数のことである。

任意の数体系の一般的な表現は，次のようになる。

$$N=d_nR^n+\cdots+d_3R^3+d_2R^2+d_1R^1+d_0R^0$$

ここで，N は与えられた数，d_n はその n 番目の位置の数字，R は与えられた数体系の底すなわち基数である。添字および R のベキ乗の数は数字の桁位置を示している。

たとえば，10 進法で 1257 というのは，次のようになる。

$$1257=1\times10^3+2\times10^2+5\times10^1+7\times10^0$$
$$(N=d_3R^3+d_2R^2+d_1R^1+d_0R^0)$$

3-2 2 進数体系（2 進法）

10 進法だけで話を進めているのなら，数体系の一般的な表現法を考えてもあまり意味がない。ここで，最も小さい底を持ち，役に立つ数体系と考えられる 2 進数体系についてその定義を与えよう。2 進法は 0 と 1 だけでできていることに注意しよう（底というのは，その数体系が用いられる数字ではなく，用いられる数字の個数である）。0 進法というのはまったく存在せず，1 進法というのは 0 という数字のみでできていることがわかる（別にとりたてて言うような数体系ではない）。ということは，2 の底を持つ数体系が，有用な数体系のうち，最も小さな底を持つ数体系であることを示している。次の対照表は 2 進数について考える出発点になるであろう。

$10^0=1$	（一）	$2^0=1$	（一）
$10^1=10$	（十）	$2^1=2$	（二）

$10^2=100$　（百）　$2^2=4$　（四）

$10^3=1000$　（千）　$2^3=8$　（八）

…　…

2進数は0と1だけでできているので，この数体系の一般的な定義は次のようになる。

$$N=\cdots+d_3\times8+d_2\times4+d_1\times2+d_0\times1$$

ここで，d_3, d_2, d_1, d_0 は，0または1である。たとえば，1011は，$1\times2^3+0\times2^2+1\times2^1+1\times2^0=8+0+2+1=11$ となる。8, 0, 2, 1 および 11 は普通の10進法で書いた数である。表 3-1 は，10進法の0から9までの数を2進法で表現したものである。

表　3-1

10 進法	2 進法	
0	0	
1	1	(1×2^0)
2	10	$(1\times2^1+0\times2^0)$
3	11	$(1\times2^1+1\times2^0)$
4	100	$(1\times2^2+0\times2^1+0\times2^0)$
5	101	$(1\times2^2+0\times2^1+1\times2^0)$
6	110	$(1\times2^2+1\times2^1+0\times2^0)$
7	111	$(1\times2^2+1\times2^1+1\times2^0)$
8	1000	$(1\times2^3+0\times2^2+0\times2^1+0\times2^0)$
9	1001	$(1\times2^3+0\times2^2+0\times2^1+1\times2^0)$

10進法なら1桁の数字だけで表わされている数を2進法で表わすと，最大4つも数字が必要で，書く場所がむだであることは明らかであるが，2進法は0と1という簡単な数字で作られていることから,大きな用途（特に計算機で）がある。数字が2つしかないので，計算機内部では,スィッチがON (1) であるかまたはOFF (0) であるか，電球がついている (1) かまたは消えている (0)か，真空管が導通の状態 (1) であるかまたは遮断の状態 (0) であるか，トランジスタがON (1) かOFF (0) か，電圧がある (1) かまたは零ボルト (0) かなどということで，2つの数字（0と1）を表わすことができる。2進法が電子計算機内部でどう使われるかということはあとの章にまわすことにして，ここでは，2進数体系とその算術演算についてだけ考える。次の2つの例を参考にして，その下に続く問題を解いてみるとよい。

例 3-1 2進数 101101 を 10 進数に書き直しなさい。

解： $N=1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$

$=1\times32+0\times16+1\times8+1\times4+0\times2+1\times1$

$=32+8+4+1=45$ (10 進数)

例 3-2 2進数 1101100 を 10 進法で書きなさい。

解： $N=1\times2^6+1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+0\times2^0$

$=1\times64+1\times32+0\times16+1\times8+1\times4+0\times2+0\times1$

$=64+32+8+4=108$ (10 進数)

問題 3-1 次の2進数を 10 進数に直しなさい。

1. 1101101　2. 10111　3. 01011　4. 1101　5. 111011101

2進数を 10 進数に直すのは，きわめて簡単であることがわかった。10 進数を2進数に直すのはどうであろうか。2576 (10 進数) は2進数でいくらか？この問題は直接的には解けないのではなかろうか。これは，次の方法を使えばできる。$(26)_{10}$——10 進法での 26 (10 進数 26) の意味——は，次のように変換できる。

		商	余り	
$\frac{26}{2}$	→	13	0	—— LSD
$\frac{13}{2}$	→	6	1	
$\frac{6}{2}$	→	3	0	
$\frac{3}{2}$	→	1	1	
$\frac{1}{2}$	→	0	1	—— MSD ——→ 11010

最後の余りを**最上位の数字**（**MSD**——most significant digit），最初の余りを**最下位の数字**（**LSD**——least significant digit）として，余りを並べる。

答：$(26)_{10}=(11010)_2$

上の答が本当かどうか逆に変換して調べてみよう。

$$N=1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0$$
$$=16+8+2=(26)_{10}$$

2で割って，その余りを逆順にとっていくことを続けるのが簡単な 10 進-2進変換の方法である。10 進法から任意の数体系へ変換するときも，同様に，その数体系の底で割っていき，その余りを逆の順に並べると答になるというこ

とがすぐにわかるであろう。次に，10進-2進変換の例をあげておくので，これを参考にしながら問題を解いてみよう。

例　3-3　$(35)_{10}$ を2進数に直しなさい。

		商	余り
$\frac{35}{2}$	→	17	1
$\frac{17}{2}$	→	8	1
$\frac{8}{2}$	→	4	0
$\frac{4}{2}$	→	2	0
$\frac{2}{2}$	→	1	0
$\frac{1}{2}$	→	0	1 → 100011

答：$(35)_{10}=(100011)_2$

検算：$N=1\times2^5+0\times2^4+0\times2^3+0\times2^2+1\times2^1+1\times2^0$

$=32+2+1=(35)_{10}$

例　3-4　$(353)_{10}$ を2進数に変換しなさい。

解：

		商	余り
$\frac{353}{2}$	→	176	1
$\frac{176}{2}$	→	88	0
$\frac{88}{2}$	→	44	0
$\frac{44}{2}$	→	22	0
$\frac{22}{2}$	→	11	0
$\frac{11}{2}$	→	5	1
$\frac{5}{2}$	→	2	1
$\frac{2}{2}$	→	1	0
$\frac{1}{2}$	→	0	1 → 101100001

答：$(353)_{10}=(101100001)_2$

問題 3-2

1．$(37)_{10}=(?)_2$　　2．$(49)_{10}=(?)_2$　　3．$(85)_{10}=(?)_2$
4．$(100)_{10}=(?)_2$　　5．$(557)_{10}=(?)_2$

2 進小数　いままで，2 進数のうち整数（integer）について考えてきたので，今度は小数（fractional number）について考えよう。10 進法では，0.5176 は，5 個の十分の一と 1 個の百分の一と，7 個の千分の一と，6 個の万分の一とから成るものである。最初の桁位置（小数点以下の）は，十分の一である。一般の数体系でも，小数は次のような意味を持っている。

$$N=d_1\times R^{-1}+d_2\times R^{-2}+d_3\times R^{-3}+\cdots+d_n\times R^{-n}$$

例 3-5　0.725 は次のものを意味する。

$$N=7\times 10^{-1}+2\times 10^{-2}+5\times 10^{-3}$$
$$(N=d_1\times R^{-1}+d_2\times R^{-2}+d_3\times R^{-3})$$

2 進小数 0.1011 は

$$1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}+1\times 2^{-4}$$

2^{-1}，2^{-2} などは 10 進数で言うと，

$$2^{-1}=\frac{1}{2^1}=0.5,\quad 2^{-2}=\frac{1}{2^2}=0.25,\quad 2^{-3}=\frac{1}{2^3}=0.125,\quad 2^{-4}=\frac{1}{2^4}=0,0625$$

であるから，

$$(0.1011)_2=(0.5+0.125+0.0625)_{10}$$
$$=(0.6875)_{10}$$

次に 2 つの例題をあげておこう。

例 3-6　$(0.101101)_2$ を 10 進数に変換しなさい。

解：$(0.101101)_2=1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}+1\times 2^{-4}+0\times 2^{-5}+1\times 2^{-6}$
$=0.5+0.125+0.0625+0.015625$
$=(0.703125)_{10}$

例 3-7　$(0.10001)_2$ を 10 進数に直しなさい。

解：$(0.10001)_2=1\times 2^{-1}+0\times 2^{-2}+0\times 2^{-3}+0\times 2^{-4}+1\times 2^{-5}$
$=0.5+0.03125$
$=(0.53125)_{10}$

問題 3-3　次の 2 進小数を 10 進小数に直しなさい。

1．0.11011　　2．0.01010　　3．0.00101
4．0.11101　　5．0.01110

2 進小数を 10 進数にするのは以上のように直接的で，容易にできるが，そ

の逆はどうであろうか。10 進の 0.57251 を 2 進数に直すのは，一見，大変なようであるがそうでもない。次のような簡単なやり方でできる。

0.57251	0.14502	0.29004	0.58008	0.16016	
× 2	× 2	× 2	× 2	× 2	
1.14502	0.29004	0.58008	1.16016	0.32032	…
↓	↓	↓	↓	↓	
1	0	0	1	0	

答は，$(0.10010\cdots)_2$ である。

正しいかどうか検算してみよう。

$$
\begin{aligned}
(0.10010)_2 &= 1\times2^{-1}+0\times2^{-2}+0\times2^{-3}+1\times2^{-4}\\
&= 0.5+0.0625\\
&= (0.5625)_{10}
\end{aligned}
$$

上に得られた数は完全にはもとの数と等しくないが，2 進数の桁数をさらに右へ右へと増していけば，与えられた 10 進数にどんどん近くなる。もし与えられた 10 進数が，$\frac{1}{2}$，$\frac{1}{8}$，$\frac{1}{16}$ など，またはこれらの和であったら，変換した 2 進数は，有限の長さになる。2 進数が有限桁でなくても，桁数を増して精度を上げることは容易にできる。次に 2 進小数が有限の長さになるものと，そうでないものの，2 つの例をあげる。

例　3-8　$(0.65625)_{10}$ を 2 進小数に直しなさい。

答：0.10101

検算：

$$
\begin{aligned}
(0.10101)_2 &= 1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+0\times2^{-4}+1\times2^{-5}\\
&= 0.5+0.125+0.03125\\
&= (0.65625)_{10}
\end{aligned}
$$

例　3-9　$(0.8176)_{10}$ を 2 進小数に直しなさい。

解：

0.8176	0.6352	0.2704	0.5408	0.0816	0.1632	
× 2	× 2	× 2	× 2	× 2	× 2	
1.6352	1.2704	0.5408	1.0816	0.1632	0.3264	…
↓	↓	↓	↓	↓	↓	
1	1	0	1	0	0	

答：$(0.110100\cdots)_2$

検算：$(0.110100)_2 = 1\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}+0\times2^{-5}+0\times2^{-6}$

$= 0.5+0.25+0.0625$

$= 0.8125$

この答が精度として十分でないと思ったら，さらにこの操作を続ければよい。

問題 3-4 次の 10 進-2 進変換の問題を解いてみなさい。桁数が大きくなりそうなときは，小数以下 8 桁まで求めればよい。

1. $(0.7257)_{10}$ 2. $(0.2501)_{10}$ 3. $(0.001876)_{10}$
4. $(0.9765)_{10}$ 5. $(0.734375)_{10}$

整数部と小数部の両方がある 2 進数を 10 進数にするのは，すぐにできる。たとえば $(11010.10110)_2$ は，次のようになる。

$$N = 1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$
$$+1\times2^{-4}+0\times2^{-5}$$
$$= 16+8+2+0.5+0.125+0.0625$$
$$= (26.6875)_{10}$$

例 3-10 2 進数 10110.1101 を 10 進数に変換しなさい。

解：$(10110.1101)_2 = 1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}$

$+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}$

$= 16+4+2+0.5+0.25+0.0625$

$= (22.8125)_{10}$

例 3-11 $(101101.110001)_2 = (?)_{10}$

解：$(101101.110001)_2 = 1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0+1$

$\times2^{-1}+1\times2^{-2}+0\times2^{-3}+0\times2^{-4}+0\times2^{-5}+1\times2^{-6}$

$= 32+8+4+1+0.5+0.25+0.015625$

$= (45.765625)_{10}$

問題 3-5 次の 2 進数を 10 進数に直しなさい。

1. $(10111.011)_2$ 2. $(1011.101)_2$ 3. $(11011.111)_2$
4. $(110.0111)_2$ 5. $(111011.001101)_2$

今度は $(274.1875)_{10}$ を 2 進数に変換する場合を考えよう。これは，数の整数部に対しては，2 で割ってその余りを逆順にとっていくことを繰り返し，小数部に対しては，2 を掛けて桁あふれの部分をとっていけばできる。

例　3-12　$(274.1875)_{10}=(?)_2$

整数部：$(100010010)_2$

小数部：$(0.0011)_2$

答：$(274.1875)_{10}=(100010010.0011)_2$

問題 3-6

1. $(27.75)_{10}=(?)_2$　2. $(37.875)_{10}=(?)_2$　3. $(521.1875)_{10}=(?)_2$
4. $(259.498)_{10}=(?)_2$　5. $(32.32)_{10}=(?)_2$

3-3　8 進法およびその他の数体系

2 進-10 進変換および 10 進-2 進変換の方法を他の数体系に対しても適用することができる。次の 2 つの例は，基本的な規則がどのように適用されているかを示すものである。

例　3-13　5 進数 124 を 10 進数に変換しなさい。

解： 底5の数体系では，次のような対応表ができる。

5 進法		10進法	
5^1	5^0	10^1	10^0
0	0	0	0
0	1	0	1
0	2	0	2
0	3	0	3
0	4	0	4
1	0	0	5
1	1	0	6
1	2	0	7
1	3	0	8
1	4	0	9
2	0	1	0
…		…	

この対応表を頭に入れて，次のように変換ができる。

$$
\begin{aligned}
(124)_5 &= 1\times5^2+2\times5^1+4\times5^0 \\
&= 25+10+4 \\
&= (39)_{10}
\end{aligned}
$$

例 3-14 8進数 376 を 10 進数に変換しなさい。

解： 8進法の各桁は，$8^0=1$，$8^1=8$，$8^2=64, \cdots$ を意味しており，8進数と10 進数との対応は次のようになる。

8 進法		10進法	
8^1	8^0	10^1	10^0
0	0	0	0
0	1	0	1
0	2	0	2
0	3	0	3
0	4	0	4
0	5	0	5
0	6	0	6
0	7	0	7
1	0	0	8
1	1	0	9
1	2	1	0
…		…	

$$\begin{aligned}(376)_8 &= 3\times 8^2+7\times 8^1+6\times 8^0\\ &=192+56+6\\ &=(254)_{10}\end{aligned}$$

問題 3-7

1. $(376)_8=(\ ?\ )_{10}$　　2. $(256)_8=(\ ?\ )_{10}$　　3. $(143)_5=(\ ?\ )_{10}$
4. $(1212)_3=(\ ?\ )_{10}$　　5. $(666)_7=(\ ?\ )_{10}$

10 進法から他の数体系への変換は，整数部に関しては除算を繰り返し，小数部に関しては，乗算を繰り返せばよい。8 進数では，次の例のようになる。

例　3-15　$(127)_{10}=(\ ?\ )_8$

解：

余り

$$\frac{127}{8}=15\cdots 7$$

$$\frac{15}{8}=1\cdots 7$$

$$\frac{1}{8}=0\cdots 1 \longrightarrow 177$$

答：$(177)_8$

検算：$(177)_8=1\times 8^2+7\times 8^1+7\times 8^0$

$$=64+56+7$$

$$=(127)_{10}$$

問題 3-8

1. $(139)_{10}=(\ ?\ )_8$　　2. $(2137)_{10}=(\ ?\ )_8$　　3. $(12)_{10}=(\ ?\ )_8$
4. $(675)_{10}=(\ ?\ )_5$　　5. $(95)_{10}=(\ ?\ )_3$

計算機において，8 進数は実用的に見て重要であるから，8 進-2 進変換について述べておこう。これは実に簡単にできる。この変換が簡単なために，8 進法がたいへん有用になるのである。まず，最初にいままでと同様の技法で変換を行なってみよう。

例 3-16 $(275)_8$ を2進数に直しなさい。

$$
\begin{aligned}
\frac{275}{2} &= 136 \cdots 1 \\
\frac{136}{2} &= 57 \cdots 0 \\
\frac{57}{2} &= 27 \cdots 1 \\
\frac{27}{2} &= 13 \cdots 1 \\
\frac{13}{2} &= 5 \cdots 1 \\
\frac{5}{2} &= 2 \cdots 1 \\
\frac{2}{2} &= 1 \cdots 0 \\
\frac{1}{2} &= 0 \cdots 1 \longrightarrow 10111101
\end{aligned}
$$

答：$(10111101)_2$

ここで次のような疑問を持つかもしれない。割算がまちがっているのではないか，それなのに答は正しい（検算してみるとよい)。いままでのやり方は誤りなのではないだろうか。いや，そうではない。方法はよいのである。疑問を持ったとすれば，多分，割算は8進数で行なわなければならないということを見落としていたからであろう。たとえば，275 を2で割るということは，まず2を2で割って商 1，7 を2で割って商 3，余り 1，次に 15 を2で割ることになるのだが，これは，8進数の15を8進数の2で割ることになる（$(15)_8$ は，10 進数で 13 であることに注意)。そのために，これは商 6，余り1 となる（商 7，余り1ではない)。前に簡単な変換法があると言ったが，上の方法は明らかに簡単だとは言えない。次に，同じ問題でいかに簡単になるかを示そう。

$$
(275)_8 = \underbrace{010}_{\substack{\downarrow \\ \text{2進法の2}}} \quad \underbrace{111}_{\substack{\downarrow \\ \text{2進法の7}}} \quad \underbrace{101}_{\substack{\downarrow \\ \text{2進法の5}}}
$$

各桁の数を2進法で表わして，そのまま並べた 010 111 101 が答である。

前の割算による答と同じになった。この手順は，8進数の各桁をそれぞれ3桁の2進数に直したことになっている。8進法で使う数字は0から7までであり，また，3桁の2進数で0から7まで扱えるから，両者はきれいに一対一に対応がつく。この2つの数体系間の変換がどのぐらい簡単かを，実際に変換して確かめてみよう。

$$(3576)_8=(011\ 101\ 111\ 110)_2$$
$$=(011101111110)_2$$
$$(2412)_8=(010\ 100\ 001\ 010)_2$$
$$=(010100001010)_2$$

次の変換はどうなるだろうか？

$$(3978)_8=(\ ?\ )_2$$

これはできるはずがない！この数は 8 進数ではない。8 進数は，高々 7 までの数字しかないから，この数を表わすことができるのは，もっと底が大きい体系でなければならない。

問題 3-9

1．$(261)_8=(\ ?\ )_2$　2．$(372)_8=(\ ?\ )_2$　3．$(42176)_8=(\ ?\ )_2$
4．$(25)_8=(\ ?\ )_2$　5．$(1376)_8=(\ ?\ )_2$

2 進数から 8 進数への変換も簡単である。次に 2 つの例と問題をあげておく。

例 3-17　$(101110110)_2=(\ ?\ )_8$

解：$(101110110)_2=(566)_8$

例 3-18　$(1011011110)_2=(\ ?\ )_8$

解：この問題では，数字を秩序正しく読むために，数字を少しつけ足す必要がある。右の方から 3 つずつ集めて組を作っていく。数字が左の方になくて組が作れなかったら，左に 0 をつけ足す。

1 011 011 110 は，001 011 011 110 となる。

すなわち，$(1336)_8$ となる。101 101 111 0 と分けてはいけない。このようなまちがいをおかさないように注意する必要がある。

問題 3-10

1．$(101111110)_2=(\ ?\ )_8$　2．$(101101)_2=(\ ?\ )_8$　3．$(11101110)_2=(\ ?\ )_8$
4．$(101101111)_2=(\ ?\ )_8$　5．$(11101)_2=(\ ?\ )_8$

2 進法の数を読むとき，長くなると数を認識するのがむずかしくなる。たとえば，10111101110111 などという数を読んでみなさい。同じものを 8 進法で書くと，$(27567)_8$ となり，憶えやすく，また見やすい。計算機での演算は 2 進法で行なわれ，答を印刷するときに，結果を 8 進数でとり出すことができる。そうすれば，これを操作している人間（オペレータ）は 2 進法で印刷されるよりも読みやすくなる。もちろん，我々が慣れている 10 進法で出てくれば，よりよいことは確かであるが，これを実現するには，2 進法の「言語」から外部へ 10 進法で出すめんどうな操作が必要で，時間がかかり，特殊な回路が必要

である。この機能を備えた計算機もあるが，簡単な8進数への変換を用いるものや2進数でしかとり出せないものもある。どちらを選ぶかは使用者の意志によるが，2進–10進変換が必要になったら，そのときには，プログラムで処理したり，またはそのため付加回路を取り付けたりすることができる。

3-4 2進法の算術

計算機で使われている2進法というのが，どういうものかを述べてきたが，今度は，2進法でどのように基本的な算術演算が行なわれるかについて考えてみよう。基本的な算術というのは，もちろん，加算，減算（引き算），乗算（掛け算），除算（割り算）の四則演算である。これらの演算は 10 進法の場合と同じであるが，2進法の構造が簡単なため演算の手順も簡素化されている。4つの演算を鉛筆と紙を使って行なってみよう。実際の計算機内部での演算は，動作するのに最良の案を選んで回路を組むために，多少様子がかわるが，これについては演算装置の章で述べることにする。

2進法の加算 2進法の加算は，次の加算表（表 3-2）から知ることができる。10 進法の加算より，簡単なことは明らかである。

表 3-2 加算表

	+	0	1	
		加数		
被加数	0	0	1	和
	1	1	$0+c$	

注： c は桁上りを意味する

この表によると，わずか4つの組合せを「記憶」するだけでよいことがわかるが，10 進法では 100 組（多くは，同じようなものの繰り返しであるが）も「記憶」しなければならない。$0+0$，$1+0$，$0+1$ は明らかである。$1+1$ という加算をすると，2になるが，2進法では，これは2桁の数になる。2進法では，各桁ごとに考えられる最大の数または数字は1である。10 進法では，これは9である。それで，$1+1$ は0と**桁上り**（carry）とになる。桁上りは，もちろん次の桁へ加えることになる。次に加算の例をいくつかあげておこう。

例 3-19

$$\begin{array}{r} 001101 \\ +\ 100101 \\ \hline 110010 \end{array} \qquad \left(\begin{array}{r} 13 \\ +\ 37 \\ \hline 50 \end{array}\right)_{10}$$

例 3-20

$$\begin{array}{r} 1011011 \\ +\ 1011010 \\ \hline 10110101 \end{array} \qquad \left(\begin{array}{r} 91 \\ +\ 90 \\ \hline 181 \end{array}\right)_{10}$$

例 3-21

$$\begin{array}{r} 110111011 \\ +\ 100111011 \\ \hline 1011110110 \end{array} \qquad \left(\begin{array}{r} 443 \\ +\ 315 \\ \hline 758 \end{array}\right)_{10}$$

問題 3-11 次の計算をしなさい。

1. $\begin{array}{r} 01101 \\ +\ 10101 \\ \hline \end{array}$　2. $\begin{array}{r} 11011 \\ +\ 10110 \\ \hline \end{array}$　3. $\begin{array}{r} 1100011 \\ +\ 1011011 \\ \hline \end{array}$

4. $\begin{array}{r} 1001101111 \\ +\ 1011100110 \\ \hline \end{array}$　5. $\begin{array}{r} 101010101 \\ +\ 111111111 \\ \hline \end{array}$

2 進法の減算　2 進法の減算も，10 進法のときと同じ操作で行なわれる。加算のときよりはめんどうであると思うかもしれないが，10 進法の減算ができるならば，もっと簡単に 2 進法の減算を理解することができる。10 進法での減算は次のようにする。

$$\begin{array}{rl} 1572 & （被減数） \\ -\ 964 & （減　数） \\ \hline 608 & （差） \end{array}$$

1572 を被減数，964 を減数とした減算を行なうと，差 608 が得られる。2 進法での減算表は，表 3-3 のようになる。

表 3-3 減 算 表

	−	0（被減数）	1（被減数）	
減数	0	0	1	差
減数	1	$1+b$	0	差

注：b は借り（borrow）を意味する。

2 進法での減算は上の減算表を用いて容易に行なうことができる。次に 2 つの例をあげておこう。

例 3-22

$$\begin{array}{r} 10110 \\ -\ 01010 \\ \hline 01100 \end{array} \qquad \left(\begin{array}{r} 22 \\ -\ 10 \\ \hline 12 \end{array}\right)_{10}$$

例 3-23

$$\begin{array}{r} 11011001 \\ -\ 10101011 \\ \hline 00101110 \end{array} \qquad \left(\begin{array}{r} 217 \\ -\ 171 \\ \hline 46 \end{array}\right)_{10}$$

問題 3-12

1. $\begin{array}{r} 110101 \\ -\ 101010 \\ \hline \end{array}$　2. $\begin{array}{r} 1101101 \\ -\ 0011110 \\ \hline \end{array}$　3. $\begin{array}{r} 10111010 \\ -\ 01111011 \\ \hline \end{array}$

4. $\begin{array}{r} 11011101 \\ -\ 10111110 \\ \hline \end{array}$　5. $\begin{array}{r} 10101010 \\ -\ 01010101 \\ \hline \end{array}$

減算を簡単にする1つの方法は，数を分けて読むことである。この方法は，役に立つことが多い。その例をあげてみよう。

例 3-24

$$\begin{array}{r} 100110011101 \\ -\ 010101110010 \\ \hline \end{array} \Longrightarrow \begin{array}{ccc} 1001 & 1001 & 1101 \\ 0101 & 0111 & 0010 \\ \hline \end{array}$$

$$\Downarrow$$

$$\begin{array}{ccc} 1001 & 1001 & 1101 \\ 0101 & 0111 & 0010 \\ \hline 0100 & 0010 & 1011 \end{array}$$

答：010000101011

減算をうまくやるもう1つの技法は 補数を使うことである。10 進法の場合に補数を用いることから話を進めてもよいのだが，2 進法の減算において主にその価値があるのだし，混乱を避けるため，2 進法の場合についてだけ説明することにしよう。はじめに，2つの定義を下しておこう。N を2進数とし，n を自然数とするとき，

　2^n-N を，n 桁の2進数 N の **2 の補数** (TWO's complement) と言い，

　$(2^n-N)-1$ を，n 桁の2進数 N の **1 の補数** (ONE's complement) と言う。

1の補数は，2の補数より1だけ小さい。逆に，2の補数は，1の補数より1だけ大きい。

例　3-25　$N=101101$ の 2 の補数を求めなさい。

$N=45$　　$n=6$（6 桁）

$2^n=2^6=64$

$2^n-N=64-45=19=010011$

2 進法の減算を使えば，

$N=101101$

$2^n=2^6=1000000$

$$
\begin{array}{r}
2^n-N=1000000 \\
-\ 101101 \\
\hline
0010011
\end{array}
$$

また，1 の補数は，

$$
\begin{array}{r}
010011 \\
-\ \ 1 \\
\hline
010010
\end{array}
$$

減算を簡単にするために補数を用いようというはずだったが，補数を求めるのに減算を使っているのでは，普通の減算と同じくらいめんどうである。話をはっきりさせていこう。2 進法で，1 の補数を求めることは，実は，機械的な操作でできるのである。例 3-25 では，N は 101101 で，N の 1 の補数は 010010 となっている。この 2 つの数を注意してみると，次のことがわかる。すなわち，与えられた N の，どの桁もすべて反対（0→1，1→0）にすると，1 の補数になるということである。これは，偶然の一致ではなく，どんな場合にも成り立ち，このため，補数を使うことが容易になるのである。

例　3-26　$N=$ 0110110101

1 の補数　:　　1001001010

2 の補数　:　　1001001011

問題 3-13　次の 2 進数の 1 の補数と 2 の補数を求めなさい。

1. 0110101　　2. 011110111　　3. 110110111

さて，補数を求めるのが非常に簡単であることがわかったので，今度はこの補数で減算をするにはどうしたらよいかを考えてみよう。減算は被減数に減数の 2 の補数を加えることによって行なうことができる。

例　3-27

$$
\begin{array}{rl}
1011011 & N_1 \\
-\ 0101110 & N_2 \\
\hline
0101101 &
\end{array}
\Longrightarrow
\begin{array}{rl}
1011011 & N_1 \\
+\ 1010010 & (2^n-N_2) \\
\hline
\text{無視する} \rightarrow (1)0101101 &
\end{array}
\qquad
\left(\begin{array}{r}
91 \\
-\ 46 \\
\hline
45
\end{array}\right)_{10}
$$

上の例で注意すべきことは，減算する 2 つの数 N_1 と N_2 が同じ桁数であり，補数を使って得た答は，やはり前の2つと同じ桁数だけが採用され，左にとび出した1は無視することである。

例　3-28

$$\begin{array}{r} 11011011 \\ -\ \ 10111 \\ \hline 11000100 \end{array} \Rightarrow \begin{array}{r} 11011011 \\ -\ 00010111 \\ \hline \end{array} \Rightarrow \begin{array}{r} 11011011 \\ +\ 11101001 \\ \hline 1\ \underbrace{11000100} \end{array} \quad \left(\begin{array}{r} 219 \\ -\ \ 23 \\ \hline 196 \end{array} \right)_{10}$$

上の例では，まず，桁数が同じになるように減数の上位に 000 をつけ加え，補数をとって，それを被減数に加えて，答として出てきた結果のうち，左にとび出した1を無視するという操作をしている。さて，補数（2の補数）を使うということには，1の補数を求める操作が含まれているから，次にそれについて考えてみよう。

計算機では，1の補数は簡単に作ることができる。0 を1に，1 を0にするだけでよい。この操作を「補数をとる」と言おう。以後，単に補数と言った場合には1の補数を意味し，2 の補数をとるときは特別にことわることにする。減算に使われているのは，2の補数であり，これは補数に1を加えたものである。よって，補数を用いて計算した結果に1を加えても同じことになる。減数が被減数より大きいことも小さいこともあるので，例をあげて，減算の操作を正確に示すことにしよう。

例　3-29

$$\begin{array}{r} 1110111 \\ -\ 0101101 \\ \hline \end{array} \Rightarrow \begin{array}{r} 1110111 \\ +\ 1010010 \\ \hline (1)\ 1001001 \\ \llcorner\!\longrightarrow 1 \\ \hline 1001010 \end{array} \quad 1\ \text{加える} \qquad \left(\begin{array}{r} 119 \\ -\ \ 45 \\ \hline 74 \end{array} \right)_{10}$$

左にとび出した1が現われ，かつ，答に 1 を足さなければならないから，このとび出した 1 を答の最下位の桁に加え込めばよい。この1のことを，**循環桁上り** (end arround carry) という。これは 1 の補数を使ったときにだけ行なわれる桁上りである。左へとび出す1がないと，循環桁上りはないことになるが，この場合は，答は負の数であるとみなされる（この数は補数の形をしている)。この数の補数（1 の）をとり，前に負の符号をつけて読む。たとえば，次のようである。

例　3-30

$$\begin{array}{r} 1011101 \\ -\ 1101100 \\ \hline \end{array} \Longrightarrow \begin{array}{r} 1011101 \\ +\ 0010011 \\ \hline 1110000 \end{array} \quad \left(\begin{array}{r} 93 \\ -\ 108 \\ \hline -\ 15 \end{array}\right)_{10}$$

答：　-0001111

左にとび出した 1 がないので，循環桁上りは必要ない。答は前に負の符号をつけ，補数をとって読むことになる。

例　3-31

$$\begin{array}{r} 10111011 \\ -\ 01100110 \\ \hline \end{array} \Longrightarrow \begin{array}{r} 10111011 \\ +\ 10011001 \\ \hline (1)\ 01010100 \\ \llcorner\!\longrightarrow 1 \\ \hline 01010101 \end{array} \text{循環桁上り} \quad \left(\begin{array}{r} 187 \\ -\ 102 \\ \hline 85 \end{array}\right)_{10}$$

答：　01010101

例　3-32

$$\begin{array}{r} 101110100111 \\ -\ 110110110111 \\ \hline \end{array} \Longrightarrow \begin{array}{r} 101110100111 \\ +\ 001001001000 \\ \hline 110111101111 \end{array} \quad \left(\begin{array}{r} 2983 \\ -\ 3511 \\ \hline -\ 528 \end{array}\right)_{10}$$

↑
桁上りの 1 がない。答は負である。

答：　-001000010000

問題 3-14　補数を使って次の計算をしなさい。

1. $\begin{array}{r} 110111 \\ -\ 101101 \\ \hline \end{array}$　2. $\begin{array}{r} 110101 \\ -\ 101101 \\ \hline \end{array}$　3. $\begin{array}{r} 1101101 \\ -\ 1110000 \\ \hline \end{array}$

4. $\begin{array}{r} 1011101 \\ -\ 1001111 \\ \hline \end{array}$　5. $\begin{array}{r} 10111101 \\ -\ 11000011 \\ \hline \end{array}$

このように，2 進数の減算は減数の 1 の補数を加えることによって行なえる。もっと複雑な減算も加算におきかえて同様に差を求めることができる。

2 進法の乗算　2 進法の乗算は，あらゆる体系のうち，最もやさしいであろう。これは，乗数の数字が，0 と 1 しかないことを考えれば明らかである。作られていく部分積は，0 か被乗数そのものになる。次の例題を見てみよう。

例　3-33

```
    110101 —— 被乗数        ⎛    53 ⎞
  ×    111 —— 乗　数        ⎜ ×   7 ⎟
 ──────────                 ⎜───────⎟
    110101                  ⎝   371 ⎠10
   110101
  110101
 ──────────
 101110011 —— 積
```

計算機では，乗算は加算を繰り返して行なう。すなわち，最終的な積を求めるために，部分的な積を作り順次加え合わせていくのである。この手順の詳細は，演算の手順を知れば，完全に了解できる。部分積を作ることは，非常に簡単である。むしろ，できた部分積を加えることの方が，桁上りを生じるからかえってめんどうである。この手続きを習得するのには，実際に計算してみるのがよい。たくさんの数の加算では桁上りがたくさん出てくるが，その扱い方について，読者がよく理解できるように，桁上りのある大きな数の乗算の例を次に示しておく。

例　3-34　次の2進法の乗算をしなさい。

```
          110110111    ⎛   439 ⎞
       ×    1010111    ⎜ ×  87 ⎟
      ─────────────    ⎜───────⎟
          110110111    ⎝ 38193 ⎠10
         110110111
        110110111
       000000000
      110110111
     000000000
    110110111
   ─────────────────
    1001010100110001
```

多くの数の加算を行なうとき，便利な方法は，1 つの桁の列の中に，1 および桁上りがいくつあるか数えることである。もしそれが偶数個ならば，和は0で，奇数個ならば和は1である。次の，上の桁へ上がるべき桁上りがいくつあるかを決めるには，その桁にある1および桁上りの1について，2 つずつ組にし，その組の数を数える。この手順は機械的であるから，理解しやすいであろう（当然誤りも少なくなる）。

問題 3-15　次の計算をしなさい。

1. $\begin{array}{r} 1011 \\ \times\ \ 110 \\ \hline \end{array}$　　2. $\begin{array}{r} 1001 \\ \times\ 1011 \\ \hline \end{array}$　　3. $\begin{array}{r} 111 \\ \times\ 100 \\ \hline \end{array}$　　4. $\begin{array}{r} 1101 \\ \times\ \ 101 \\ \hline \end{array}$

5. $\begin{array}{r} 1001111 \\ \times\ \ \ 11101 \\ \hline \end{array}$

2 進法の除算　乗算と同様に、除算も簡単にできる。除算の途中で中間の被除数の中に除数が入っているかどうかだけ見ていけばよいのであって，商の各桁は1か0である。

例 3-35

$$\begin{array}{rrl} & 1100 & \text{（商）} \\ \text{（除数）}\ 110 & \overline{)1001000} & \text{（被除数）} \\ & -110 & \\ \cline{2-2} & 00110 & \\ & -\ 110 & \\ \cline{2-2} & 0000000 & \end{array} \qquad \left(6\overline{)72}^{\;12}\right)_{10}$$

まず，除数と同じ桁数だけ被除数の桁を頭からとり，割ってみる（100 を 110 で割る）。もし割れなかったら，さらに1桁被除数の桁を増して，同じことを行なう（1001 を 110 で割る）。割ることができれば，商に1が立つ。除算の手順は，10 進法の場合よりも簡単である。

問題 3-16　次の計算をしなさい。

1. $101\overline{)11001}$　　2. $11\overline{)10010}$　　3. $1001\overline{)1010001}$
4. $1010\overline{)1100100}$　　5. $1101\overline{)100111}$

要約

2進法の数体系を導入して，整数および小数を 10 進から2進に，また，逆に2進から 10 進に変換することを述べた。計算機は，その内部では，すべて2進法か2進コードの形で動くので，2 進法での算術は計数型計算機の基本的な動作を理解するのに必要なものである。また，多くの計算機の言語は，8 進法の形で表現されるから，8 進法も大切である。すなわち，演算コードやオペランド（被演算子）は8進数で表わされ，計算機からの直接的な出力が8進数で表示されることもある。前にも述べたように，10 進法で入力や出力を行なうことができない機械ではプログラムで変換すればよい。

2進法の減算を行なうために，補数を使うことは，多くの計算機でこれを採用しており，大切なことである。減算については，主に，1の補数の加算と循環桁上りを加えて計算する方法を述べたが，2の補数による方法も計算機にはよく採用されている。この方法では，ある特殊な計算機回路で直接に2の補数をとり出し，それを被減算数に加えて減算を行なう。

2進法の乗算や除算は加算や減算を繰り返して行なえばよい。計算機の内部で行なう方法については，10章の演算装置の説明のところで詳しく述べる。

問題

1. 次の2進数を10進数に直しなさい。

(a) 110111 (b) 111000 (c) 010101
(d) 101010 (e) 1111110

2. 次の10進数を2進数に直しなさい。

(a) 25 (b) 67 (c) 99 (d) 135 (e) 276

3. 次の2進小数を10進小数に直しなさい。

(a) 0.1010 (b) 0.11 (c) 0.001 (d) 0.011 (e) 0.11001

4. 次の10進小数を2進小数に直しなさい。

(a) 0.8750 (b) 0.0930 (c) 0.370 (d) 0.53125 (e) 0.4375

5. 次の10進－8進変換をしなさい。

(a) $(45)_8=(\ ?\ )_{10}$ (b) $(63)_{10}=(\ ?\ )_8$ (c) $(125.3)_8=(\ ?\ )_{10}$
(d) $(119)_{10}=(\ ?\ )_8$ (e) $(625.5)_{10}=(\ ?\ )_8$

6. 次の2進－8進変換をしなさい。

(a) $(1101101)_2=(\ ?\ )_8$ (b) $(372)_8=(\ ?\ )_2$ (c) $(101110.111)_2=(\ ?\ )_8$
(d) $(2753)_8=(\ ?\ )_2$ (e) $(25.57)_8=(\ ?\ )_2$

7. 次の計算を，示された数体系で行ないなさい。

(a) $\left(\begin{array}{r}111011\\+\ 110\\\hline\end{array}\right)_2$ (b) $\left(\begin{array}{r}111110111\\+\ 111001\\\hline\end{array}\right)_2$ (c) $\left(\begin{array}{r}365\\+\ 23\\\hline\end{array}\right)_8$

(d) $\left(\begin{array}{r}2732\\+\ 1265\\\hline\end{array}\right)_8$ (e) $\left(\begin{array}{r}10111\\11011\\+\ 10111\\\hline\end{array}\right)_2$

8. 示された数体系で，次の減算をしなさい。

(a) $\left(\begin{array}{r}11001\\-\ 110\\\hline\end{array}\right)_2$ (b) $\left(\begin{array}{r}5372\\-\ 2561\\\hline\end{array}\right)_8$ (c) $\left(\begin{array}{r}375\\-\ 127\\\hline\end{array}\right)_8$

(d) $\left(\begin{array}{r} 11101101 \\ -\ 01001111 \\ \hline \end{array}\right)_2$ (e) $\left(\begin{array}{r} 11011011110 \\ -\ 10100111001 \\ \hline \end{array}\right)_2$

9. 示された数体系に従い，次の乗算と除算を行ないなさい。

(a) $\left(\begin{array}{r} 1011 \\ \times\ \ 11 \\ \hline \end{array}\right)_2$ (b) $\left(\begin{array}{r} 2325 \\ \times\ \ 23 \\ \hline \end{array}\right)_8$ (c) $\left(\begin{array}{r} 110 \\ \times\ 11011 \\ \hline \end{array}\right)_2$

(d) $(1010\overline{)1100100})_2$ (e) $(24\overline{)740})_8$

10. 補数を用いて，次の2進法の減算をしなさい。各計算について1の補数の方法，2の補数の方法を示しなさい。

(a) $\begin{array}{r} 1011 \\ -\ 0011 \\ \hline \end{array}$ (b) $\begin{array}{r} 10101 \\ -\ 01100 \\ \hline \end{array}$ (c) $\begin{array}{r} 101010 \\ -\ 010111 \\ \hline \end{array}$

(d) $\begin{array}{r} 100001 \\ -\ 11110 \\ \hline \end{array}$ (e) $\begin{array}{r} 1011011101100 \\ -\ 1100101101111 \\ \hline \end{array}$

0100＝4

コード

情報をコード化 (coding) するということは，他の記号を使って文字（たとえば数字とか英字）を表わすことである。コードは昔から他の人が文を理解できないように秘密保持のために使われてきた。しかし，計算機では，2 進符号 1 と 0 で文字を表わしており，多くの種類の 2 進コードが目的に応じて選択されて使われている。ある種のコードは算術演算を行なう場合に使われる。他のコードには，より少ない**ビット**（bit, ビットは 2 進数 1 桁，1 または 0）で，より多くの情報を与えるという点で高い効率を持つものがある。誤りの検出または訂正ができるコードは重要なものとされている。計算機はコード化されて送られた文字が正しく受けとられたかどうかを判断し，もし誤りがあれば訂正することができなくてはならない。コード化することそれ自身深く研究されている問題だが，ここでは 2，3 のよく使われているコードについてだけ述べる。

4-1 2 進化 10 進数

基本的なコードは **2 進化 10 進数** (binary-coded decimal)，または **BCD** と呼ばれるものである。このコードでは，10 進数 0～9 を表わすのに 2 進数を使う。BCD の数は 1 と 0 を使って書かれているので，これは一種のコードである。表 4-1 に BCD コードを示してある。

BCD コードは 1 つの 10 進数を表わすのに，4 桁（4 ビット）の 2 進数が必要である。明らかに，このコードは 10 進数よりはるかに能率が悪い。しかし，それは計算機の内部で使われているのと同じ 1 と 0 で書かれているという利点がある。このコードで数がどのように表わされるかを例をあげて示す。

表 4-1 BCD コード

10 進数	2 進化 10 進数
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

10 進	BCD
22	0010 0010
35	0011 0101
671	0110 0111 0001
2579	0010 0101 0111 1001

例でわかるように，各 10 進数字はそれに対応して，コード化された 4 ビットの 2 進数を必要とする。BCD コードは 10 進数と比べると数を表わすためにより多くの桁を必要とする。しかし，これは 2 進記法で表わしてあるので非常に便利である。もう 1 つ注意すべき点は，4 つのビットの内での各ビットの場所が重要であるということである。その場所に重みをつけることがあり，しばしばコードの形を述べるためにそれが使われる。第 1 の場所（右端）には 2^0 すなわち 1，第 2 には 2^1 すなわち 2，第 3 には 2^2 すなわち 4，第 4 には 2^3 すなわち 8 の重みをつける。左から読むと 8-4-2-1 である。そのため BCD コードは 8-4-2-1 コードとも呼ばれる。

このコード（8-4-2-1）が 2 進数と異なることをはっきりさせるため，次のようなことを考えよう。すなわち，十の 2 進数表示は 1010，2 進化 10 進数で表わすと 0001 0000 である。十六は 2 進数 10000 であり，8-4-2-1（BCD）コードは 0001 0110 である。両者の間の混乱は最初の 9 までの数が BCD でも 2 進数でも同じであるということから生じる。それ以後は両者はまったく異なる。次の問題でためしてみよう。

問題 4-1 次の 10 進数を BCD コードで書きなさい。
1．275　2．362　3．9256　4．100　5．2790

BCD コードの主要な価値は，8 進数と同様に，見やすいことである。たとえば，2 進数と BCD で書かれた数のどちらが読みやすいか，実際に比較してみるとよくわかるだろう。

10 進	2 進	BCD
141	10001101	0001 0100 0001
2179	100010000011	0010 0001 0111 1001

算術演算でこのコード化された形を使うとなると問題が生じてくる。8 に 7 を加える計算を 2 進数と BCD について行なってみよう。

10 進	2 進	BCD	
8	1000	1000	
+ 7	+ 0111	+ 0111	
15	1111	1111	← これは BCD の文字にはない。(15 は 0001 0101 である。)

BCD コードで演算するためには，特別な加算器が必要となる。読みやすいという性質が必要とされ，また算術演算も重要な場合には，修正したコードを使うことがある。

4-2 3 あまりコード

BCD コードを修正したものに **3 あまりコード** (excess-three code) があるが，これを 表 4-2 に示してある。BCD コードと 3 あまりコードを注意深く比較してみれば，その相違点と，どのように 3 あまりコードが作られたかが明らかになるだろう。その名が表わすように，3 あまりコードでコード化された

表 4-2 3 あまりコード

10 進	BCD	3 あまりコード
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

文字は，BCD のそれより3大きい。6すなわち 0110 は，1001 と書かれる。BCD では，1001 は9だが，3 あまりコードでは6なのである。“3 あまり” ということを忘れてはいけない。2,3 のコードの例をあげる。

10 進	3 あまりコード
2	0101
25	0101 1000
629	1001 0101 1100
3271	0110 0101 1010 0100

算術演算で，3 あまりコードがいかに有効かを見るため，次の例を考えてみよう。

```
  3           0110
+ 9           1100
----         ------
 12         1 0010
     +  0011  0011
     -------------
        0100  0101
```

答は 0100 0101（3 あまりコード）すなわち 12 である。

加算にはいくつかの特別な規則（例に示したように各数に3を加えるなど）がある。しかし，これらの段階は計算機の中では自動的に行なわれるし，そのための回路を作ることも容易である。そのため3あまりコードは算術演算に適している。各桁から3を引かなければならないため，読みにくくなっているが，大きな数の2進表示そのままよりはまだ読みやすい。

問題 4-2 次の数を 3 あまりコードで書きなさい。

1．279　2．301　3．2176　4．1568　5．21769

BCD コードは重みのついたコードであったが，3 あまりコードはそうではない。BCD で第2（2^1）の場所のビットは2を意味する。3 あまりコードでは，どの場所のビットも特定の数の加算を意味しない。たとえば，BCD の 0100 は4で，それに 2^1 ビットをつけ加えて 0110 にすると，2 だけ大きい数6になる。3 あまりでは，0111 は 4，1001 は6であり，各ビットに対応した重みはない。

4-3 2-5 進コード

重みのついたコードに，**2-5 進コード**（biquinary code）と呼ばれるものがある。コードは7ビットから成り，2 ビットと5ビットに分かれている。コー

ドには重みがついている。0〜9のコード化された形と各ビットの重みを表4-3に示す。

表 4-3

10 進	2-5 進コード
	50　43210
0	01　00001
1	01　00010
2	01　00100
3	01　01000
4	01　10000
5	10　00001
6	10　00010
7	10　00100
8	10　01000
9	10　10000

このコードは10進数を表わすのに7ビット（BCDや3あまりコードでは4ビット）必要である。2-5進法の重要な特徴として，コードの語に誤りがあれば，そのことがわかる構造になっていることをあげることができる。情報がある場所から他の場所へ送られるとき，たとえば計算機の装置から装置へ，空中から地上へ，地上局から地上局へ送られるときなど，誤りが発生したかどうか調べることのできるコードは大変有用である。表4-3の2-5進コードを注意深く見ると，各語に2つだけ1があることがわかる。もし余分に1が現われれば，どこかにまちがいがあるので，その語は受け入れることができないことを示している。もし1が1つしかない場合も，誤りであることは明らかである。さらに語を正しく読みとるためには，最初の2ビットと次の5ビットに，それぞれ1が1つずつなくてはならない。このことを調べる回路を作るのは簡単なので，容易に検査機構を確立することができる。

例 4-1　次の2-5進コードの誤りを探しなさい。

	10 進	2-5 進
(a)	4	01 10001
(b)	5	01 10010
(c)	6	10 10101
(d)	6	11 00010
(e)	31	01 01000　01 00010

(f)　99　10 10000　10 10000

(g)　0　01 00001

答：例 (a)-(d) は誤り，(e)-(g) は正しい。

問題 4-3　次の数を 2-5 進コードで書きなさい。

1. 56　2. 731　3. 68　4. 732　5. 509

4-4 コードのパリティ

誤りの検出と訂正は，ディジタルデータの伝送において，その研究と応用が発展しつつある分野である。誤りを検出するために非常によく使われる方法は**パリティ** (parity) ビットを使用する方法である。たとえば，穿孔紙テープはテープから計算機への読み込みや，その逆の操作での正確さを増すために，パリティ検出機構を持っている。パリティは偶数，奇数どちらでもよい。パリティビット (2 進の 1 または 0) を加えることにより，その文字のコードの 1 のビットの総数を偶数または奇数にする。説明のために，BCD コードにパリティビットを加えてみる。このビットは 2^0 の桁の右につけ加えられる。偶数パリティビットをつけ加えることにより，1 のビットの数を偶数にする。奇数パリティでは，1 のビットの数が奇数となるように選ばれる。コード化された文字が受信されると，パリティを検査する（前もって偶数か奇数か選んでおく)。検査に合格すれば，正しいものとして受け入れられる。表 4-4 は BCD コードと偶数パリティのついた BCD，奇数パリティのついた BCD を示す。パリティの型によって，パリティビットが逆になっているのがわかる。

表 4-4　BCD コードにおけるパリティ

10 進	BCD コード	BCD（奇数パリティ）	BCD（偶数パリティ）
0	0000	00001	00000
1	0001	00010	00011
2	0010	00100	00101
3	0011	00111	00110
4	0100	01000	01001
5	0101	01011	01010
6	0110	01101	01100
7	0111	01110	01111
8	1000	10000	10001
9	1001	10011	10010

例　4-2　次のパリティビットのついたBCDの文字が正しいかどうか調べなさい。

	語	パリティビット	パリティの型
(a)	1001	0	奇数
(b)	1000	0	奇数
(c)	0001	0	偶数
(d)	1010	0	偶数
(e)	0110	1	奇数

答：(a) と (c) の例は誤り，(b) と (e) は正しい。(d) は BCD コードの文字(数字)でないので誤り。しかし偶数パリティとしてのパリティビットは正しい。パリティは BCD コードの場合以外でも使われる。種々の語を送るときも，パリティビットを同じようにつけることがある。

例　4-3　次の例に誤りがあるかどうか調べなさい。

	語	パリティビット	パリティの型
(a)	0110111101	1	偶数
(b)	1101110100	0	奇数
(c)	1110111011	0	奇数
(d)	1011011100	0	偶数
(e)	1010111010	1	奇数

答：(b) と (c) は誤り。(a), (d) および (e) は正しい。

BCD コードに戻って正しい例をあげておこう。

(f)　10 進 57 は 01011 01110 (奇数パリティ)

(g)　10 進 79 は 01111 10010 (偶数パリティ)

問題 4-4　指定したパリティを持つ BCD コードを書きなさい。

1. 10 進数　95 (偶数パリティ)
2. 10 進数 7986 (奇数パリティ)
3. 10 進数　605 (奇数パリティ)
4. 10 進数　71 (偶数パリティ)
5. 10 進数 6094 (奇数パリティ)

4-5 グレイ コード

グレイコード(Gray code) は光学的または機械的**軸位置符号器** (encoder) に

広く使われている。これは重みづけのないコードで，主な特徴は続いている文字の間で1つのビットしか変化しないということである。これは円周上に順に異なった語が対応するようなコード **ホィール*** で使われている。グレイコードは1つの桁だけのあいまいさを持っている。この特徴は入力装置についての章（12 章）でさらに述べる。ここではコード自身が主題である。

1語あたり 数 ビットを持ち，一度に1ビットずつしか変化しないコードとしては多くのものがあるが，そのうち“グレイ”と呼ばれるコードをなぜ選んだかということは興味のあることである。入力装置のコードの型の決定は，読取りの精度を上げそして製作上の問題を単純にするために重要なことである。計算機の内部でこのコードを使用することは，重みがついていないため，非常に困難である。そのため，計算機の内部で重みのついたコード（たとえば完全な2進数）に容易に変換できるならば,このコードを用いることは都合がよい。グレイ コード は外見としては1ビットしかかわらないということと, 2進の形に変換するのが簡単であるという点で有効さを持つ特別なコードである。

10 進の 0-12 に対し，2 進数と グレイ コード** の対応表を 表 4-5 に示してある。すべての数について グレイ コード があるので，表に示したのは単なる例にすぎない。

表 4-5　10 進数 0-12 に対するグレイコード

10 進数	2 進数	グレイコード
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010

* code wheel

** 〔訳注〕 最大数から 0 に戻るときも，1 ビットしか変化しないようになっている。

グレイ コード では，続いている2つの語の間で 1ビットしか変化していないことに注意しなさい。2 進数ではそのようなことはない。10 進数の7から8に進むとき，2 進数では4ビット全部がかわるが，グレイコードでは1つしかかわらない。9から 10 になるときも，2 進数では 1001 から 1010 にかわる。2^0 ビットは1から0になり，2^1 ビットは0から1になり，2 つのビットがかわる。グレイ コード では，1101 から 1111 と変化し，2^1 ビットの0が1に1つ変化するだけである。グレイから2進へ，2 進からグレイへの変換には，はっきり定まった変換規則があるので，以下それについて考えてみよう。

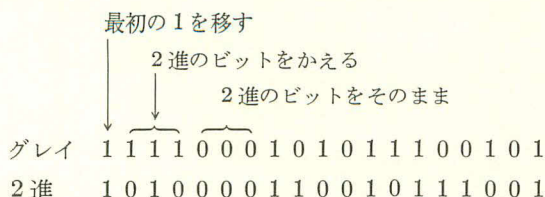
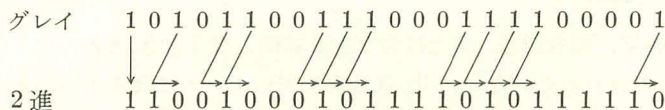
グレイコードでコード化された語を，2進コードにするには，最上位のビット (MSB, most significant bit) から始める。2 進数では，最下位のビット (LSB, least significant bit) は 2^0 ビットで，MSB は最も大きい重みを持つ場所 (4 ビットでは 2^3) である。2 進数とそれと等価なグレイコードは，

```
        MSB
グレイ   1 0 1 1 0 1 0 1 1 1 0 0 1
2  進    1 1 0 1 1 0 0 1 0 1 1 1 0
```

ここで MSB は左端である。変換するためには，グレイの最初の1を含めて1が出てきたら，それまで並べてきたビットの0と1とを切り替え，グレイのビットが0のときは以前のものを並べればよい。この例では1でスタートするので，それを2進数の MSB のビットとして用いる。同じ2進ビットを次の グレイ ビット が0である限り繰り返して続けていく (この例では1回)。グレイコード に 2 番目以降の1が出てきたら，2 進の語のそれまでのビットをかえる。2進のビットは1だったので，それは0にかわる。この規則を続けていくと，グレイのその次の1は，それまでの2進ビットの0を1にかえることになる。グレイに0が続くと，2進のビットをそのままにしておくことを意味する。例題で，グレイの2番目のビットは0であるから，2進数では1を繰り返して並べてある。このような操作を語の終りまで繰り返す。説明は複雑だが，例題をみればはっきりわかるだろう。

例 4-4

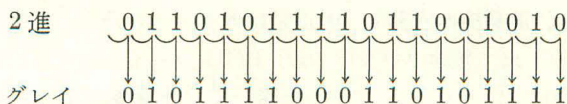
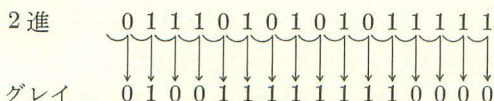
```
            2進をそのまま
               ↓  2進をかえる
                   ⌒
グレイ    0 1 0 1 1 0 0 1 1 1 1 0 1 0 1
2  進     0 1 1 0 1 1 1 0 1 0 1 1 0 0 1
     最初の1
```

例 4-5

例 4-6

もしグレイの語に1が現われると，それまでの2進ビットをかえる。グレイの0は，2 進をそのままにしておく。この変換の規則が上例でよくわかったであろう。

問題 4-5 次のグレイ コード 化された語を 2 進数の形にかえなさい。

1. 1001110101101
2. 101010101010
3. 01110111011110
4. 1001100011100001111
5. 11111111111111

2 進の語からグレイに変換する手順を述べるのはいくらかやさしい。 MSB から始めて，続いているビットの比較を行なうことによって，変換ができる。もし等しければ，グレイの語に0を書く。異なっていれば1を書く。出発点では，第1のビットを0と比較する。

例 4-7

例 4-8

注意： ⌉は2つのビットを MOD 2（桁上りなし）で加えることを意味する。そしてその結果をグレイビットとして下に書く。

問題 4-6 次の 2 進数をグレイコードに変換しなさい。

1. 0110101111011101 1
2. 10001111011010110
3. 10100101101001
4. 1111111111
5. 0010101110111

4-6 ASCII 入出力コード

英字，10 進数字，特殊記号，または命令を計算機で扱うとき，それらを 2 進数にコード化しなければならない。10 進数字を表わすためには 4 ビットのコード（たとえば BCD）が必要である。これらの 10 進数の他に 26 個の英字と若干の特殊記号を取り扱うために，少なくとも 6 ビットのコード（$2^6=64$ の組合せ）が必要である。産業界で広く採用されている標準のコードとして，**ASCII コード**（American Standards Code for Information Interchange）がある。このコードは 8 ビットコードで，小文字（lower case）と大文字（upper case）の英字と特殊記号（たとえば *，+，=）と 30 個以上の命令または制御操作（たとえば通信の始めと終り，改行復帰，行送り）を表わすことができる。

表 4-6 には 10 進数字，英字およびいくつかの特殊文字に対する ASCII コードをあげてある。ASCII コードは左から右へ読むように指定されている。同じコードを穿孔紙テープ，磁気テープ，磁気ディスク，高速印刷機，ある種のテレタイプ装置等々に用いることができる。これの例については 12 章で述べる。

8 単位のコードが入出力装置で文字を表わすために使われるが，一度計算機の中に入れば，それらのコードは異なった操作のために，より便利に取り扱うことができる。たとえば，計算機の内部で 10 進数は 8 ビットの語として扱う必要はない。4-3-2-1 ビットは，数字 0-9 のための BCD コードとしてとってある。10 進数を表わすのには，7-6-X-5 ビットを除いて，計算機は BCD の形の 4 ビットだけとっておけばよい。もし標準の語の長さが 8 ビットなら，計算機は 2 つの 4 ビットの BCD を，内部的には 1 語で取り扱うことができる。

例 4-9 2 つの ASCII の数字を 8 ビットの 1 つの語に詰めなさい。

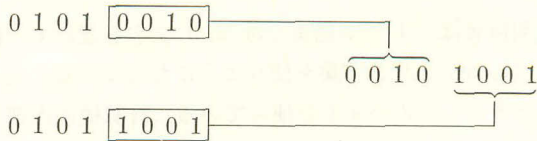
入出力装置で 01010010 01011001 と表わされた数 29 は，計算機の内部では，次のようにまとめることができる。

左側のものは，2 桁の 10 進数 29 を表わした 8 ビットの語の内容である。

表 4-6 ASCII コード

文字	7	6	5	X	4	3	2	1	文字	7	6	5	X	4	3	2	1
0	0	1	0	1	0	0	0	0	@	1	0	1	0	0	0	0	0
1	0	1	0	1	0	0	0	1	A	1	0	1	0	0	0	0	1
2	0	1	0	1	0	0	1	0	B	1	0	1	0	0	0	1	0
3	0	1	0	1	0	0	1	1	C	1	0	1	0	0	0	1	1
4	0	1	0	1	0	1	0	0	D	1	0	1	0	0	1	0	0
5	0	1	0	1	0	1	0	1	E	1	0	1	0	0	1	0	1
6	0	1	0	1	0	1	1	0	F	1	0	1	0	0	1	1	0
7	0	1	0	1	0	1	1	1	G	1	0	1	0	0	1	1	1
8	0	1	0	1	1	0	0	0	H	1	0	1	0	1	0	0	0
9	0	1	0	1	1	0	0	1	I	1	0	1	0	1	0	0	1
:	0	1	0	1	1	0	1	0	J	1	0	1	0	1	0	1	0
;	0	1	0	1	1	0	1	1	K	1	0	1	0	1	0	1	1
<	0	1	0	1	1	1	0	0	L	1	0	1	0	1	1	0	0
=	0	1	0	1	1	1	0	1	M	1	0	1	0	1	1	0	1
>	0	1	0	1	1	1	1	0	N	1	0	1	0	1	1	1	0
?	0	1	0	1	1	1	1	1	O	1	0	1	0	1	1	1	1
空白	0	1	0	0	0	0	0	0	P	1	0	1	1	0	0	0	0
!	0	1	0	0	0	0	0	1	Q	1	0	1	1	0	0	0	1
″	0	1	0	0	0	0	1	0	R	1	0	1	1	0	0	1	0
♯	0	1	0	0	0	0	1	1	S	1	0	1	1	0	0	1	1
$	0	1	0	0	0	1	0	0	T	1	0	1	1	0	1	0	0
%	0	1	0	0	0	1	0	1	U	1	0	1	1	0	1	0	1
&	0	1	0	0	0	1	1	0	V	1	0	1	1	0	1	1	0
′	0	1	0	0	0	1	1	1	W	1	0	1	1	0	1	1	1
(	0	1	0	0	1	0	0	0	X	1	0	1	1	1	0	0	0
)	0	1	0	0	1	0	0	1	Y	1	0	1	1	1	0	0	1
*	0	1	0	0	1	0	1	0	Z	1	0	1	1	1	0	1	0
+	0	1	0	0	1	0	1	1	[	1	0	1	1	1	0	1	1
,	0	1	0	0	1	1	0	0	\	1	0	1	1	1	1	0	0
−	0	1	0	0	1	1	0	1	]	1	0	1	1	1	1	0	1
.	0	1	0	0	1	1	1	0	↑	1	0	1	1	1	1	1	0
/	0	1	0	0	1	1	1	1	←	1	0	1	1	1	1	1	1

最近，計算機関係者は，1つの語または2,3文字を表わす一群のビットを呼ぶのに**バイト**（byte）という言葉を使うようになっている。たとえば，IBM SYSTEM/360は8ビットのバイトを使っている。計算機の内部でデータはビットではなくバイトを単位にして取り扱われる。8ビットのバイトは計算機の中に入ってきたままの1つのASCII文字，または，算術演算における2つの10進数字を表わすために使える。したがって，語とは，ある一定数のバイトのことである。

問題 4-7 ASCIIコードを使って，次の文字を書きなさい。

1．86　2．293　3．HELLO　4．DIGITAL　5．DIGIT NO. 3

問題 4-8 次のASCIIの文字を，1バイトあたり2つの4ビットの数にまとめ直し，等価な10進数を書きなさい。

1．01010100 01010101　2．01010011 01010110　3．01011001 01010010
4．01010001 01011000　5．01010111 01010000

要　約

この章では，一般によく使われているいくつかのコードについて述べてきた。各コードは異なる長所を持っており，それらはいろいろな分野で広く使われている。最も一般的なコードは10進数0-9を表わすために使われているBCDである。これは2進数そのものより，はるかに読むことが容易である。しかし，算術演算では，そう簡単に使うことはできない。そのため，“3あまり”コードと呼ばれる修正された形が使われる（このコードはBCDほど読みやすいものではない）。

2進データを送るとき，誤りを検出できるために，2-5進コードを使ったり，コード化された文字にパリティビットを付加したりする。多くの種類のコードが使われているが，データに対する信頼性，どれだけ余分な情報が送れるか，または検査を行なうためにどれだけ余分の設備が必要かによって，その中から選んで用いられている。

グレイコードは，コードホイール（12章参照）上のデータを表わすために使われる。コードはコードホイールまたは円盤の上ではある型でなければならず，また計算機の内部では，完全に2進数として取り扱うのが最もよいので，グレイから2進へ，2進からグレイへのコード変換が必要になる。

最後に，入出力装置への多くの文字や指令を表わすためのASCIIコードについて述べた。これについてはさらに12章で述べる。

問　題

1. $(2573)_{10}$ を BCD で書きなさい。
2. $(10)_{10}$〜$(16)_{10}$ を BCD で書きなさい。
3. $(39287)_{10}$ を BCD で書きなさい。
4. $(2718)_{10}$ を 3 あまりコードで書きなさい。
5. $(6259)_{10}$ を 2-5 進コードで書きなさい。
6. BCD 1001 1000 0101 を 2-5 進コードで書きなさい。
7. 2 進数 1101101101 をグレイ コードに変換しなさい。
8. グレイコードの 110110101 を 2 進数に変換しなさい。
9. $(3572)_{10}$ を偶数パリティつき BCD で書きなさい。
10. $(2598)_{10}$ を奇数パリティつき 3 あまりコードで書きなさい。
11. 27 を ASCII コードで書きなさい。
12. A+B=C を ASCII コードで書きなさい。

0101 — 5

ブール代数

5-1　ブール代数の基礎

ブール代数（Boolean algebra）は，論理的な問題を考えるときに用いられる数学的方法である。1847 年，イギリスの数学者ジョージ・ブール（George Boole）は，演繹的論理の問題に適するような数学の基本的な法則と規則を展開した。1938 年まで，この技法は，数学分野だけのものであったが，この年，偉大な科学者であるクロード・シャノン（Claude E. Shannon）は，この代数の性質が役に立つものであることに気づき，電話の結線の分析のためにこれを応用した（彼はベル研究所で働いていた）。計算機の発展につれて，ブール代数は電子工学の分野で盛んに応用されるようになり，いまや技術者の論理設計の助けとして必須なものになってきた。

もともと，ブール代数は，結果が真または偽である命題に関して記述するものである。シャノンは，開いているか，閉じているかのいずれかである接点を持つ回路網について調べるのに，これを用いたのである。計算機の面では，さらに，1 または 0 の状態を持つ回路網を記述するのに用いられる。論理の 1 または 0 は，2 進法の 1 または 0 に対応づけることができるし，開いているか閉じている，または，真か偽の状態にあてはめることができる。これらはすべて 2 進法の性質をもっている。この数字が 2 つの値だけをとる変数を含んでいるのであるから，ブール代数というのは，変数が連続した値をとる普通の代数学に比べたら，きわめて単純なものである。

まず，この代数の，基本的な規則，演算および恒等式をいくつか考えてみよう。まず 0 と 1 に関する演算をはっきり定義しておかなければならない。加算に似た OR の演算と，乗算に似た AND の演算がある。このような演算が，なぜ必要かは，これを実際に応用していくにつれて明らかになっていくであろ

う。ORの演算は，ORで結ぶ2つの変数がとり得るすべての組合せについて考察することによって定義することができる。すなわち，次の通りである。

$$0+0=0$$
$$0+1=1$$
$$1+0=1$$
$$1+1=1$$

注：“+”はORを意味する。

上の表の最後の演算で，1+1=1というのは，普通の意味での加算と異なっている点に注意しよう。これは，「1 OR 1は1に等しい」と読む。2人の人間が互いに関連したスイッチを操作して，1つの電球を点滅させる例で，ORの働きを説明することができる。2人とも，自分のところにあるスイッチをOFF (0)にしておけば，電球は消えて(OFF(0))いる。1人が自分のところにあるスイッチをON(1)にすれば，電球はON(1)になる。他の1人がスイッチをON(1)にしてもよい。もちろん，両方のスイッチがON(1)でも，電球はON(1)となる。このことは，2人のスイッチのORをとったものが，結果として，電球に現われていることになる。各条件の組合せに対する出力が上に示したような関係になる関数のことを，**OR関数**と呼ぶ。

ORの演算を果たすためには，スイッチを並列に結べばよい。2つのスイッチAおよびBを用いた回路が図5-1に示してあるが，このようなスイッチング回路を記述するのにブール代数のORを用いることができる。すなわち“A+B=電球への出力”である。

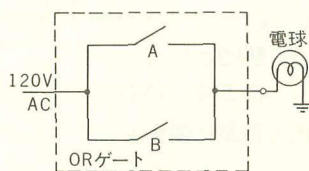



図5-1　ORゲート

スイッチが閉じている状態を状態1，スイッチが開いている状態を状態0と定め，電球については，ついている状態を状態1，消えている状態を状態0と定める。Aが閉じているか，またはBが閉じていれば，電球はついており，状態1となる。両方のスイッチがともに閉じていてもその出力は1である。ともに開いているときだけ，電球は状態0になる。すなわち，OR関数で可能な組合せをすべて表わしていることになる。図5-1の点線で囲んだ回路を，OR

ゲートという。

さて，上の回路を見ると，スイッチが並列に結線されている。当然，直列（図5-2）にすることも考えられ，これがどうなるかということに興味が出てくる。この結線は乗算に似た働きをする AND の演算の説明になるのである。AND とは，どういうものであるかを調べるには，前のように可能な組合せを作ってみればよいが，それは次のようなものである。

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

数字の間にある点は，AND の演算を示す記号である。AND の演算で，文字を用いるときは，点を省略することもある。図 5-2 のスイッチと電球の回路を調べ，この回路が **AND 関数**を表わしているかどうか，考えてみよう。

スイッチ A が閉じていて，スイッチ B が開いていると（A は 1，B は 0），電球は消えている（すなわち OFF (0)）。スイッチ B が閉じていて，A が開いていても，同じである。A と B の両方が閉じている場合にだけ電球は ON となる。OR の演算では，両方のスイッチが閉じている場合は ON であるが，どちらか一方が閉じていても ON であった。これに対し，AND の演算では，両方のスイッチが閉じている場合だけ ON となり，その他の場合はすべて OFF になる。

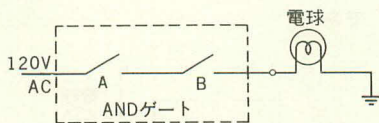



図 5-2 AND ゲート

ここで，これらの基本的な演算に関するいくつかの恒等式を考えてみよう。

$$A + 0 = A$$
$$A \cdot 0 = 0$$
$$A + 1 = 1$$
$$A \cdot 1 = A$$
$$A + A = A$$
$$A \cdot A = A$$

以上の恒等式が正しいかどうかを証明するには，A がとりうるすべての値について，この恒等式が成立するかどうかを見ればよい。A がとる値は 0 と 1 だけ

であるから，証明は簡単にできる。たとえば，A+0=A については，次のとおりである。

$A+0=A$	恒等式が正しいことを示している。
$0+0=0$	
$1+0=1$	

同様にして残りのものも証明できる。すなわち，

$A\cdot 0=0$	$A+1=1$	$A\cdot 1=A$	$A+A=A$	$A\cdot A=A$
$1\cdot 0=0$	$0+1=1$	$0\cdot 1=0$	$1+1=1$	$1\cdot 1=1$
$0\cdot 0=0$	$1+1=1$	$1\cdot 1=1$	$0+0=0$	$0\cdot 0=0$

この代数についてさらに検討を加えてみると，一般の代数の基本的な法則を満足していることがわかる。すなわち，次に述べる法則である。

交換則　$A+B=B+A$

　　　　$A\cdot B=B\cdot A$

結合則　$A+(B+C)=(A+B)+C$

　　　　$A\cdot(B\cdot C)=(A\cdot B)\cdot C$

分配則　$A\cdot(B+C)=A\cdot B+A\cdot C$

変数の可能な組合せについて演算の結果がどのようになっているかを，系統的に調べていくために，**真理値表** (truth table) というものを用いる。真理値表を用いると，いくつかの式があったとき，それらが等しいものかどうかを調べることもできる。2 つの式からそれぞれ作られた2つの真理値表を見て，あらゆる入力の組合せ（変数の値の組合せ）に対して，2 つの結果が同じになっていることがわかれば，2 つの式は等しく，また2つの式が等しければ，これらの式から作られる真理値表の同じ入力に対する値は相等しい。入力の考えられるあらゆる組合せを表にする1つの方法は，例に示してあるように，2 進数を順番に上から1つずつ増加するように書いていくことである。変数が2つある場合，考えられる組合せの総数は，$4(=2^2)$ であり，変数が 3 つの場合には $8=2^3$ である。一般には，n 変数のときには，2^n の組合せが考えられる。

代数法則を証明するとき，どのように真理値表が役立つかをみてみよう（図 5-3 参照）。図 5-3b を見ると，すべての可能な入力の組合せに対して，A+(B+C) に対する演算結果は，(A+B)+C に対するものと，同じであるから，この2つの式は同等である。この真理値表で，(A+B) と，(A+B)+C を比べてみると，ただ1個所が違っているだけであるが，これだけでも，この2つの式が異なっているというのには十分である。A•(B•C) と (A•B)•C では，

変数の可能な組合せについて式の値が同じであるから，この2つの式は同等である。図 5-3c は，ブール代数の分配則の証明のための真理値表である。

A	B	(A+B)	(B+A)
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

(A+B)=(B+A)

A	B	A・B	B・A
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

A・B=B・A

図 5-3a　交換則の真理値表（2変数）

A	B	C	(B+C)	A+(B+C)	(A+B)	(A+B)+C
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

A+(B+C)=(A+B)+C

A	B	C	(B・C)	A・(B・C)	(A・B)	(A・B)・C
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	1	0
1	1	1	1	1	1	1

A・(B・C)=(A・B)・C

図 5-3b　結合則の真理値表（3変数）

A	B	C	(B+C)	A·(B+C)	(A·B)	(A·C)	(A·B)+(A·C)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

図 5-3c　分配則の真理値表

ブール代数での変数 (Boolean variable) は, 2 つの値――たとえば H(high) と L(low), T(true) と F(false), または 0 と 1 などで表わす――のうちの 1 つの値をとることができる。次に, 変数 A の否定である NOT A を考える。文字の上にバーをつけたり後にダッシュをおいたりして, この否定関数 ($\bar{A}$ または A′) を表わすことがある。この関数に関係する簡単な恒等式をあげておこう。

$$A + A' = 1$$

$$A \cdot A' = 0$$

$$(A')' = A$$

上の等式を, 前と同じように, 真理値表で証明してみると, 次の通りである。

$A + A' = 1$	$A \cdot A' = 0$	$(A')' = A$
$0 + 1 = 1$	$0 \cdot 1 = 0$	$1 = 1$
$1 + 0 = 1$	$1 \cdot 0 = 0$	$0 = 0$

NOT の関数の特別な関係式に, **ド・モルガンの定理** (De Morgan's theorem) と呼ばれるものがある。すなわち,

$$(A+B)' = A' \cdot B'$$

または　$$(A \cdot B)' = A' + B'$$

この証明は真理値表を用いて容易にすることができる。この法則の意味は, 2 つの変数を AND または OR で結んだ式を否定すると, 先に変数を否定したものを, もとの演算子 (AND, OR) ととりかえたもの (AND ならば OR, OR ならば AND) で結んだものになる, ということである。ド・モルガンの定理の応用例を以下に示そう。

例 5-1 $A+BC$ の否定。

解： $$(A+[B\cdot C])'=A'\cdot(B\cdot C)'$$
$$=A'\cdot[B'+C']$$
$$=A'B'+A'C'$$

例 5-2 $AB+C'D'$ の否定。

解： $$(AB+C'D')'=(AB)'\cdot(C'D')'$$
$$=(A'+B')\cdot(C+D)$$
$$=A'C+B'C+A'D+B'D$$
$$=C(A'+B')+D(A'+B')$$

例 5-3 $A'B'C'+A+B+C$ の否定。

解： $$(A'B'C'+A+B+C)'=(A'B'C')'\cdot A'\cdot B'\cdot C'$$
$$=(AA')B'C'+(BB')A'C'+(CC')A'B'$$
$$=0\cdot B'C'+0\cdot A'C'+0\cdot A'B'$$
$$=0+0+0$$
$$=0$$

最後の例は，与えられた式が，いままでに与えてある恒等式を用いて簡素化できることを示している。ここに例示した 式の簡素化 という点で，ブール代数は論理設計にたいへん役立っているのである。

例 5-4 $(AB+C)A$ を簡単にしなさい。

解： $$(AB+C)A=A\cdot AB+A\cdot C$$
$$=(A\cdot A)B+A\cdot C$$
$$=A\cdot B+A\cdot C$$
$$=A(B+C)$$

例 5-5 $A+A'B$ を簡単にしなさい。

解： 式は，簡単になっているが，もっと簡素化できる。
$$A+A'B=A(B+B')+A'B$$
$B+B'=1$ であるから，A に $B+B'$ を掛けても値はかわらない。展開すると，
$$A(B+B')+A'B=AB+AB'+A'B$$
この式に AB を加えても，値はかわらない。なぜなら，
$$AB+AB=AB$$
$$AB+AB'+A'B=AB+AB+AB'+A'B$$
これをまとめて，

$$A(B+B')+(A+A')B$$
$$=A\cdot 1+1\cdot B$$
$$=A+B$$

すなわち，$A+A'B=A+B$ である。真理値表を使って，この等式が正しいことを証明することができる。

式の変形は複雑であるが，結果は非常に役に立つことが多いから，複雑な問題の中に，$A+A'B$ の形が出てきた場合を考えて，この関係式を憶えておいた方がよいだろう。

例 5-6 $A'B+AB+A'B'$ を簡単にしなさい。

解：$A'B+AB+A'B'=A'(B+B')+AB$
$=A'\cdot 1+AB=A'+AB$
$=A'+B$（例 5-5 の法則を利用）

例 5-7 $(AB'+A'B)'$ を簡単にしなさい。

解：$(AB'+A'B)'=(AB')'(A'B)'$
$=(A'+B)(A+B')$
$=A'A+A'B'+BA+BB'$
$=0+A'B'+AB+0$
$=AB+A'B'$

問題 5-1 次の式を簡単にしなさい。

1. $AB+BC$
2. $AB'+BC+C'A$
3. $A'(BC+AB+BA')$
4. $ABC+CAB+AB+A$
5. $(A'+AB)(A'+B)$
6. $XY+YZ+Z'Y$
7. $[(X+Y)'+(X+Z)']Z$
8. $(XYZ+WX')'$
9. $UVW+XVW+YVW$
10. $BC+AD+ABCD+CDA+A'$

問題 5-2 真理値表を用いて，次の式が成り立つかどうかを調べなさい。

1. $XY+X'Y+X'Y'=X'+Y$
2. $ABC+AC+BC=A+B+C$
3. $(X'Y+Y'X)'+XY=(XY'+X'Y)'$
4. $ABD+A'B'D+AB'D'=A(B'D'+BD)$
5. $(AND)(DAN)+AN(AD)=AND$

5-2　リレー論理

OR 関数の導入のとき，実際の OR 回路を説明するのに，スイッチを並列に接続したものを使い，また，AND 関数の導入のときには，スイッチの直列接続を使った。直列，並列，または直列-並列のスイッチの接続による複雑な回路は電話回路の中に見られるが，計算機の中の実際の回路にも使われている。スイッチを**ゲート素子***として考えると，回路を簡素化するのに，ブール代数がどのように使われるかを理解できるであろう。閉じた回路を 1（電気的信号が通過する）と定義し，開いた回路を 0（電気的信号が止められる）と定義しよう。すると，スイッチング回路はブール代数の論理式（Boolean expression）で表わすことができる。次の例では，リレーの接続に新しい記号を用いて書いてある。

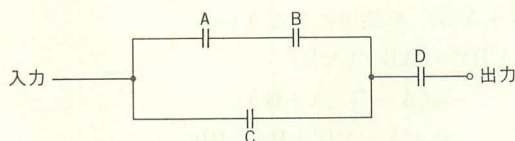



図 5-4　例 5-8 のリレー回路

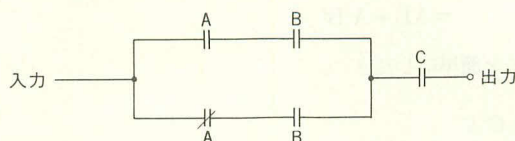



図 5-5　例 5-9 のリレー回路

例　5-8　図 5-4 の回路のブール代数式を書きなさい。

解：　出力$=(AB+C)D$

例　5-9　図 5-5 に示されている回路のブール代数式を書き，これを簡単にしなさい。

解：　出力$=(AB+A'B)\cdot C$

$=(A+A')\cdot B\cdot C$

$=1\cdot B\cdot C$

$=B\cdot C$

---

*〔訳注〕　ゲート素子とは，それに制御信号を与えることによって，入力信号を論理的に出力側に伝送するための素子である。普通ゲート素子の論理的な組合せ（論理回路）で制御回路が構成される。

上の例で，スイッチにつけた名前の中に，B という文字が 2 個所に現われている。このことは，スイッチを複数個保持して，同時に操作されるようになっていることを示している。リレーに信号が入ると，そのリレーのすべてのスイッチは動作し，各スイッチは，別々の回路または 1 つの回路の異なった部分で使われている。上の回路を簡単にすると，スイッチ B および C だけの回路におきかえることができる。回路図を見ると，A が開いていれば，A′ は閉じているのであるから，B と C を通る経路ができることは明らかである。また，A が閉じていれば，A′ は開いているのだから，これもまた，B と C の経路になる。言いかえれば，A がどのような状態になっていても，回路の状態は，B と C だけで決まる。簡単になったブール代数式は，このことを示しているのである。ブール代数式を作って簡素化するほどのことがあろうかと思うかもしれないが，電話の結線や，計算機の中の複雑な論理設計には，たいへん有用なものなのである。

問題 5-3 次の回路（図 5-6）の出力の式（ブール代数式）を書き，これを簡単にしなさい。

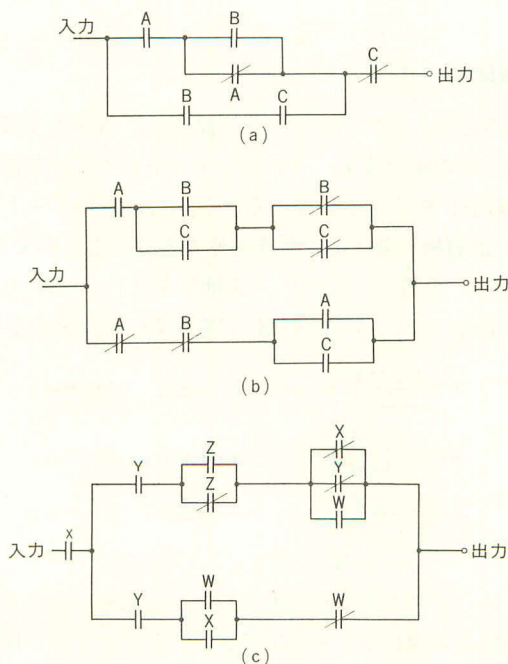



図 5-6 問題 5-3 のリレー回路（次頁に続く）

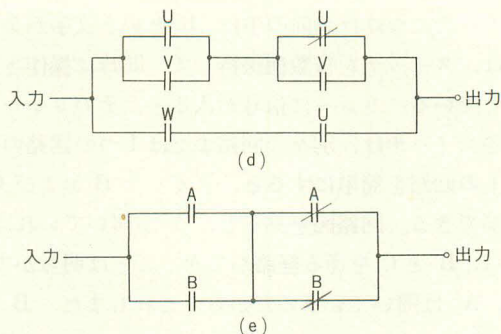



図 5-6（続き）

問題 5-4 次のブール代数式の回路図を書きなさい。式は簡単にしなくてもよい。

1. WX+WZ+XY+X′Y′
2. XUV+WX+U′V+X′
3. AB+CD+(AB)(C+D)
4. (AM+FM)(DC+AC)
5. UV+U+V+X′Y′+X′+Y′

5-3 電子式論理ゲート*

電子計算機で主に使われているゲート回路では，AND, OR，および NOT 関数，またはこれらの組合せを，トランジスタやダイオードなどの半導体素子（または，固体素子）を使って形成している。これらのゲートをつなぎ合せることによって，計算機の基本的な動作を作り上げることができる。ゲートの型を記号で示して，説明をしていく。文献により，いろいろな記号が用いられているが，この本では，ASA 規格（図 5-7）を採用することにする。ま

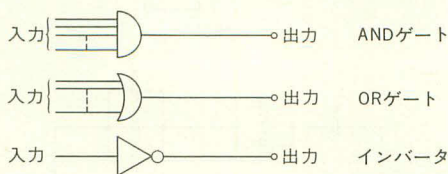



図 5-7 電子ゲートの記号

た，小さな丸で終っているところは出力であり，直線で始まるところが入力である。図上での信号の流れは，矢印で特別にことわってない限り，上から下，

*〔訳注〕 論理ゲートとは論理回路のことである。

または左から右に流れるものとする。

インバータ (inverter) は NOT の機能を果たすものである。入力に1が入ると，出力には0が現われ，入力に0が入ると，出力には1が現われる。前にも述べたとおり，これらを接続したブロックは，ブール代数式で表わすことができる。逆に，ブール代数式の意味するものを実際に果たす回路を作りあげるのに，これらを使うこともできる。

例 5-10 図 5-8 の論理図で表わされるブール代数式を書き，これを簡単にしなさい。

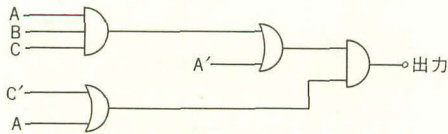



図 5-8 例 5-10 の論理図

解：出力$=(A \cdot B \cdot C+A')(A+C')$

$=AABC+AA'+ABCC'+A'C'$

$=ABC+0+AB \cdot 0+A'C'$

$=ABC+A'C'$

例 5-11 ブール代数式 $AB+AC+A'C'+B'C'$ の論理図を書きなさい。

解： 5-9 図参照。

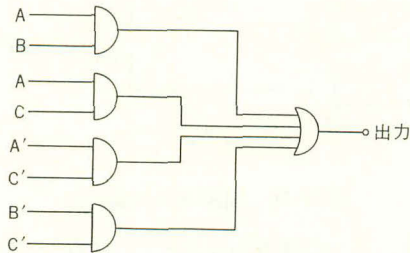



図 5-9 例 5-11 の論理図

問題 5-5 図 5-10 の論理図に対するブール代数式を書き，これを簡単にしなさい。

問題 5-6 次の式の論理図を書きなさい。簡単にしなくてもよい。

1. $(AB+CD)C'$
2. $(A'B'+CD)(AB+C)$
3. $(XY+Z)'(X'Y'Z')$
4. $WUV+W'U'V'+(UV+WX)$
5. $(X+Y)'+(X+Z) \cdot (UY) \cdot (X+Z)$

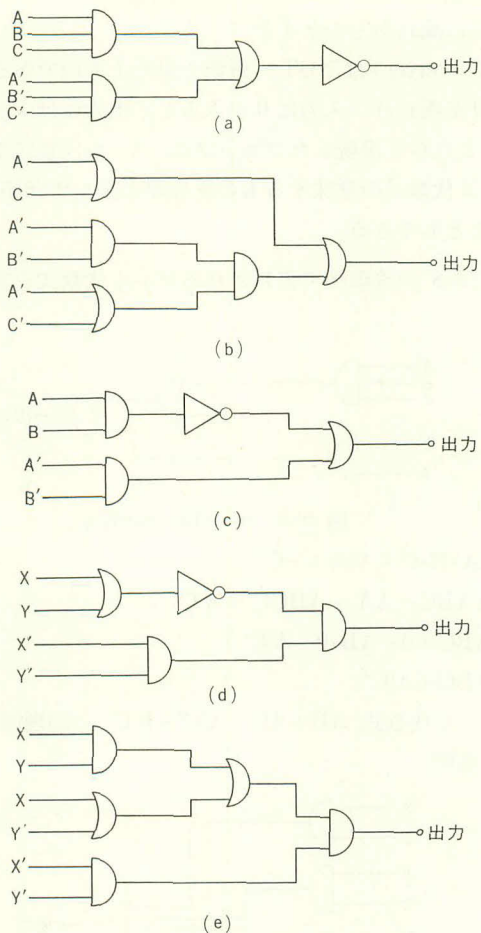



図 5-10　問題 5-5 の論理図

NAND/NOR ゲート　3つの基本的なゲート（すなわち AND, OR, INVERTER）に加えて，NAND ゲート，および NOR ゲートというものがよく使われている。AND ゲートの後に INVERTER を続けたものを NOT AND ゲートまたは NAND ゲートという。たとえば，ゲートの入力が A, B, C であると，出力が (A•B•C)′ となるゲートである。NAND ゲートの記号は，図 5-11 のように表わす。AND ゲートの後に，小さい丸をつけて，インバータの働きがあることを示している。NAND ゲートだけで，AND の働きをさせるためには，直列に2つの NAND ゲートを接続するか，または，NAND ゲ

ートの後にインバータが必要である。このような簡単な場合には，NAND を使うのは無駄なことのように見えるかもしれない。実際には，論理関数の複雑な組合せを考えるようになり，実際的な観点が重要になってくると，NANDが便利な場合が起こる。ここでは，NAND ゲートだけを使った基本的な技法を示そう。

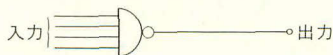



図 5-11　NAND ゲートの記号

例 5-12 NAND ゲートだけを使って，与えられた論理式の機能を果たす論理図を書きなさい。

(a)　XYZ

解：図 5-12a

図 5-12a　XYZ に対するゲートの論理図

(b)　X+Y+Z

解：図 5-12b

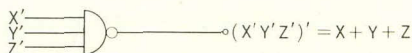



図 5-12b　X+Y+Z に対する NAND ゲートの論理図

ド・モルガンの定理により，$(X'Y'Z')'=X+Y+Z$。ド・モルガンの定理を使うと，NAND ゲートの変化に富んだ重要な使い方ができる。

(c)　$XY'+X'Z$

解：まず，NAND の型に直す。

$$(XY'+X'Z)=[(XY')'\cdot(X'Z)']'$$

右辺は，AND の項の否定でできている。こうすれば，NAND ゲートに直接に直すことができる（図 5-12c）。

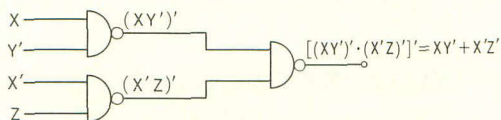



図 5-12c　XY′+X′Z に対する NAND ゲート論理図

(d)　$XYZ+Y'Z'+YZ'$

解：まず，与えられた式を，2 重否定する。こうすると，$(A')'$ が A であることから，同じものになる。

$$[(XYZ+Y'Z'+YZ')']'$$

次に，内側の NOT に対して，ド・モルガンの定理を適用する。

$$[(XYZ)'\cdot(Y'Z')'\cdot(YZ')']'$$

OR を含んでいる項から始めたが，上の 2 つの手続きで，NAND を使える形に変換できた（図 5-12d）。

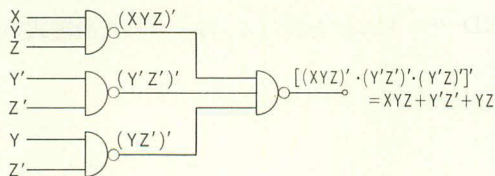



図 5-12d　$XYZ+Y'Z'+YZ'$ の NAND ゲート論理図

ここにあげた例（d）は，さらに別な形に変形できる。

$$\begin{aligned} XYZ+Y'Z'+YZ' &= XYZ+Z'(Y'+Y) \\ &= XYZ+Z'\cdot 1 \\ &= XYZ+Z' \\ &= XY+Z' \ (AB+B'=A+B' \text{ を使用}) \end{aligned}$$

これを NAND の形にすると，次のようになる（図 5-12e）。

$$\begin{aligned} XY+Z' &= [(XY+Z')']' \\ &= [(XY)'\cdot(Z')']' \\ &= [(XY)'\cdot Z]' \end{aligned}$$

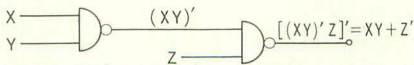



図 5-12e　$XY+Z'$ の NAND ゲート論理図

(e)　$(XY'+Z)(X'Z+Y')(XY'+Z')$

解：NAND ゲートで表示できるようにするため，まず式を展開して整理する。

$$\begin{aligned} \text{出力} &= (XY'+Z)(XX'Y'Z+X'ZZ'+Y'XY'+Y'Z') \\ &= (XY'+Z)(0+0+XY'+Y'Z') \\ &= XY'+XY'Z'+XY'Z+Y'ZZ' \end{aligned}$$

$=XY'+XY'Z'+XY'Z$

$=XY'(1+Z'+Z)$

$=XY'$ (図 5-12f 参照)

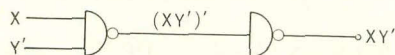



図 5-12f XY′ の NAND ゲート論理図

(f) $(XY+U)(W+XZ)(UV+Y)$

解：まず，展開する。

$(XY+U)(W+XZ)(UV+Y)=(XY+U)(UVW+YW+UVXZ+XYZ)$

$=UVWXY+UVW+XYW+UYW+UVXYZ+UVXZ+XYZ+UXYZ$

$=UVW(XY+1)+XYW+UYW+XYZ(UV+1)+UVXZ+UXYZ$

$=UVW+XYW+UYW+XYZ(1+U)+UVXZ$

$=UVW+XYW+UYW+XYZ+UVXZ$

2重否定と，ド・モルガンの定理を適用して，

$=[(UVW+XYW+UYW+XYZ+UVXZ)']'$

$=[(UVW)'\cdot(XYW)'\cdot(UYW)'\cdot(XYZ)'\cdot(UVXZ)']'$

(図 5-12g 参照)

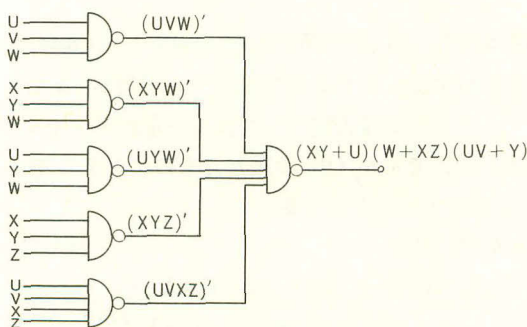



図 5-12g (XY+U)(W+XZ)(UV+Y) の NAND ゲート論理図

OR ゲートのあとに，インバータがついているものを，NOT OR または **NOR ゲート**という。NOR ゲートだけで論理式を表わすには，やはり，NAND ゲートのときと同じような考慮が必要である。NOR ゲートの記号は，OR ゲートの記号のうしろに，小さな丸をつけたものである。

例 5-13 次の論理式（ブール代数式）を，NOR ゲートだけから成る論理図で表わしなさい。

(a) $X+Y+Z$

解：図 5–13a 参照

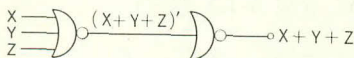



図 5-13a $X+Y+Z$ の NOR ゲート論理図

(b) XYZ

解：図 5–13b 参照

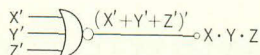



図 5-13b $X \cdot Y \cdot Z$ の NOR ゲート論理図

(c) $(X+Y')(W+Z)$

解：$(X+Y')(W+Z)=[[(X+Y')(W+Z)]']'$

$=[(X+Y')'+(W+Z)']'$

(図 5–13c 参照)

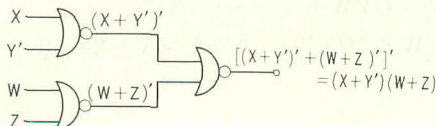



図 5-13c $(X+Y')(W+Z)$ の NOR ゲート論理図

NOR ゲートで表わすようにするときには，まず項が AND で結ばれる形に式をもっていく。これを2重否定し，内側の否定にド・モルガンの定理を使うと，各項を否定して，それを OR で結んだものになる。そして，これを外側にある NOT で否定することになる。

(d) $(X+Y+Z)(U+V)$

解：$(X+Y+Z)(U+V)=[[(X+Y+Z)(U+V)]']'$

$=[(X+Y+Z)'+(U+V)']'$

(図 5–13d 参照)

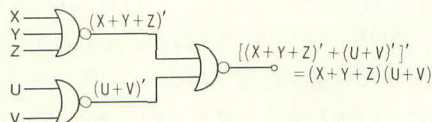



図 5-13d $(X+Y+Z)(U+V)$ の NOR ゲート論理図

(e) $XYZ+UV$

解：$XYZ+UV=[(XYZ+UV)']'$

$$=[(XYZ)'(UV)']'$$

$$=[(X'+Y'+Z')(U'+V')]'$$

$$=(X'U'+Y'U'+Z'U'+X'V'+Y'V'+Z'V')'$$

$$=[(X+U)'+(Y+U)'+(Z+U)'+(X+V)'+(Y+V)'+(Z+V)']'$$

(図 5-13e 参照)

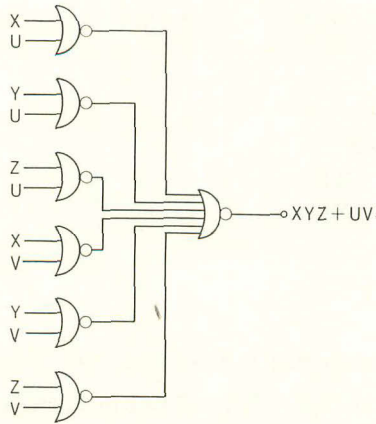



図 5-13e　XYZ+UV の NOR ゲート論理図

この式は次のようにもっと簡単に表わすことができる。

$$XYZ+UV=(X'+Y'+Z')'+(U'+V')'$$

$$=[[(X'+Y'+Z')'+(U'+V')']']'$$

(図 5-13f 参照)

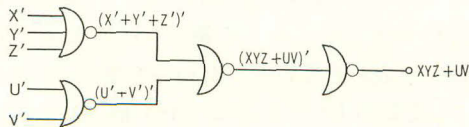



図 5-13f　XYZ+UV の NOR ゲート論理図

この例の最初の変形の方法は，直接的ではあるが，必ずしも簡単な形にしているわけではない。NAND や NOR で表現することは，実際的に見て，最も簡単な解がどれであるかを見るのによい根拠を与える。式を最も簡単なものにしたり，それが最も簡単なものであることを証明したりする定理や方法を確立するのに，たくさんの研究がされてきた。しかし，これらの技法はそれ自身複

雑なものであるから，日常作業としては向かないであろう。

問題 5-7　次の論理式を，NAND ゲートだけを使った最も簡単な論理図で表わしなさい。

1. $X'+Y+Z'$
2. $XY+XZ+YZ'$
3. $AC+BD+B'C+A'B$
4. $(XYZ)(XW)(YW)$
5. $(AB+C)(DE+F)$

問題 5-8　次の論理式を NOR ゲートだけで作りあげるための最も簡単な論理図を書きなさい。

1. $X'Y'Z'$
2. $(XY)(XZ)+YZ$
3. $X'Y'Z+XYZ+YZ'$
4. $(WX+YZ)(UV+WZ)$
5. $(XY+U)+(UV+X)$

5-4　カルノー図

カルノー図 (Karnaugh map) というのは，ブール代数の式を系統的に簡単にしていくための1つの方法である。この方法では変数がいくつあってもよいが，実際には変数が6つ以上ということはほとんどない。ここでは，最大4つの変数までの場合について考えてみよう。原理的には真理値表によく似ているが，使う場合には，まったく異なった使い方をするものである。この図（マップ）は，箱（またはエリア）と呼ばれるものの集まったものでできていて，それぞれは変数のとりうる可能な組合せの1つを表わす。変数が1つの場合には，2つの箱（Aと A′ に対応する）があればよい。変数が2つの場合には，組合せは4つ（2^2 個）あるから，4つの箱があればよく，各箱はそれぞれ，AB, AB′, A′B, A′B′ を表わしている。3つの変数のときは，2^3 すなわち，8個の箱が必要であり，変数が4つならば 2^4 (16) 個の箱が必要である。以上の4つの場合について，マップの形を 図5-14 に示す。

A	0	1

A \ B	0	1
0		
1		

A \ C, B	0 0	0 1	1 1	1 0
0				
1				

AB \ CD	0 0	0 1	1 1	1 0
0 0				
0 1				
1 1				
1 0				

図 5-14 1~4 変数に対するカルノー図

3 変数および 4 変数の場合には，必要なすべての組合せを表わすために，図 5-14 のように，行または列に 2 つの変数を同居させて，2 つの変数のとりうる組合せを列挙すればよい。このマップの使い方は次のようである。組合せ，すなわち，変数の積が式に現われたら，これに対応する箱の中に 1 を置く。たとえば，式の中に AB というのが現われたら，マップの組合せ AB に対応する箱に 1 を置く（例 5-14 参照）。この仕事がすべて終了したら，このマップ全体を見て，最も簡単な式を作り出していく。

例 5-14 次の論理式をマップに表わしなさい。

(a) $AB+A'B'$

解：図 5-15a 参照

B \ A	0	1
0	1 —— A′B′	
1		1 —— AB

図 5-15a $AB+A'B'$ のカルノー図

(b) $ABC+A'B'C'+ABC'+AB'C+A'BC$

解：図 5-15b 参照

(c) $ABCD+A'BCD+AB'CD+ABCD'$

解：図 5-15c 参照

図 5-15b　ABC+A′B′C′+ABC′+AB′C+A′BC のカルノー図

CD ＼ AB	00	01	11	10
0 0				
0 1				
1 1		1	1	1
1 0			1	

(ABCD, AB′CD, ABCD′, A′BCD)

図 5-15c　ABCD+A′BCD+AB′CD+ABCD′ のカルノー図

(d)　AB+A′C

解：図 5-15d 参照

図 5-15d　AB+A′C のカルノー図

3変数のマップでは，1つの箱は3変数の積の1つの項に対応するので，2変数の積でできた項を書き込むには，1を2つ使わなければならない。2変数でできている項は2つの箱を用いて，一義的に定めることができる。また，4つの箱を用いて，1変数の項を定義することができる。ここで，次のことに注意しておこう。2変数の項はすべて2つの箱に対応づけられるけれども，2つの箱の組合せのすべてが2変数の項を定義するわけではない。2つの箱は隣合っているなど適当な条件を満たしていなければならない。

(e) B+AC+A′C′

解：図 5-15e 参照

図 5-15e B+AC+A′C′ のカルノー図

この例でわかる重要な点は，1つの箱がいくつかの項に対応するということである。このようなことが起こっているときには，箱をまとめ直して，もっと簡単にすることができることになる。

(f) AB+AC+AB′C′ をマップに表わし，集めかえをやってみよう。

解：図 5-15f 参照

C \ AB	00	01	11	10
0			1	1
1			1	1

図 5-15f AB+AC+AB′C′ のカルノー図

上の4つの1を1つの集まりとして見ると，マップは論理式Aと読むことができる。すなわち，AB+AC+AB′C′=A である。式からの変形がうまくでき

ないとしても，マップを見れば，明らかに簡素化が存在するかどうかがわかる。この例では，代数的にすぐ簡素化ができるが，マップを使った方がより簡単である。代数的に簡単にすると，次のようになる。

$$\begin{aligned} AB+AC+AB'C' &= A(B+C+B'C') \\ &= A(B+C+C') && [B+B'C'=B+C'] \\ &= A(B+1) && [C+C'=1] \\ &= A && [B+1=1] \end{aligned}$$

(g)　カルノー図を使って，次の式を簡単にしなさい。

$$XY+XZY+X'Y+XZY'$$

解：図 5-15g 参照

図 5-15g　XY+XZY+X′Y+XZY′ のカルノー図

簡素化するために，図 5-15h のように集めかえる。

簡素化した式は，

$$XY+XZY+X'Y+XZY'=Y+XZ$$

Z ＼ X Y	0 0	0 1	1 1	1 0
0		1	1	
1		1	1	1

（グループ：Y，XZ）

図 5-15h　Y+XZ のカルノー図

(h)　1が隣合っていれば，それらを一緒にすることができるが，離れていても一緒にしていい場合がある。たとえば，図 5-15i のような場合である(B′C)。

図 5-15i B′C のカルノー図

(i) 斜めに並んでいる1は集めることができない。たとえば，図 5-15j のマップは，4 つの項から成り，それぞれを表わすのに，3 変数の項が必要である。これ以外に集めようがないわけである。

C \ AB	00	01	11	10	
0	1		1		ABC′
1		1		1	AB′C
	A′B′C′	A′BC			

図 5-15j ABC′+AB′C+A′BC+A′B′C のカルノー図

(j) 4変数のマップでは，1 変数の項は 8 つの箱，2 変数は 4 つ，3 変数は 2 つの箱を必要とし，4 変数の項は 1 つの箱で表わされる。ABC′D′+AB′D+A′Cのマップを作りなさい。

解：図 5-15k 参照

CD \ AB	00	01	11	10	
00			1		ABC′D′
01				1	AB′D
11	1	1		1	
10	1	1			
	A′C				

図 5-15k ABC′D′+AB′D+A′C のカルノー図

問題 5-9　次の論理式をカルノー図に表わして，もし簡単になりそうだったら，マップから簡単な式を導きなさい。

1. $AB+BC+AC$
2. $ABC+A'B'C'+BC'+B'C$
3. $XY+X'Z+Y'Z'$
4. $WXYZ+W'X'YZ'+WYZ'+YX$
5. $UVX+UVX'+U'X+UV'$
6. $AB+B'C+A'B'$
7. $AC+AC'B+BC+B'C'$
8. $A+CD+AC'D+A'C'D'$
9. $XY+YZ+XZ+X'Y'$
10. $UV+UX+XV+U'X'$

これと似た方法で，ブール代数の式を簡単にするのに，ヴィーチ（Veitch）図というものがある。3 変数のマップを 図5-16 に示す。これは，AB の項を 2 進数として見て，左から右に，0 から始まって，1 ずつ増加するように書かれているだけの違いであるが，多くの変数の項の組合せを作るときは，系統的で便利である。しかし，カルノー図の方が見て簡単化するのがよりやさしいように思えるので，これからの例題にはもっぱらカルノー図を使用する。

図 5-16　3 変数のヴィーチ図

5-5　論理技法の応用

この章で述べた技法を実際の問題にどのように応用するかを，2 進法での加算器および減算器について，詳しく調べてみることにしよう。2 つの 2 進数の加算を行なった結果の和を作ることを考えるために，まず，真理値表を作ってみる。2 つの入力についてだけの加算をするものを **半加算器**（half adder）という。完全な加算を行なうためには，下の桁からくる桁上りも考慮する必要があるわけで，このような場合には 3 つの入力が必要になるが，これを**全加算器**（full adder）という。半加算器の真理値表を 図5-17 に示す。

A （入力1）	B （入力2）	S （和）	C （桁上り）
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

図 5-17　半加算器の真理値表

A=0，B=1 および A=1，B=0 に対しては，和は S=0+1=1+0=1。
A=0，B=0 ならば，和は S=0+0=0。
A=1，B=1 ならば，和は S=1+1=0（そして桁上りが起こり，そのことを表わす変数 C は値1をとる。C=1)。

Sが1の値をとるのは，A=0 すなわち A′=1 でしかも B=1 のとき，または A=1 でしかも B=0 すなわち B′=1 のときである。ブール代数式で表わせば，

$$S=A'B+AB'$$

となる。このような半加算器のための回路図は，図 5-18 のようになる。

また桁上りが起こって，C=1 となるのは，A=1 でしかも B=1 のときであって，

$$C=AB$$

と表わすことができる。

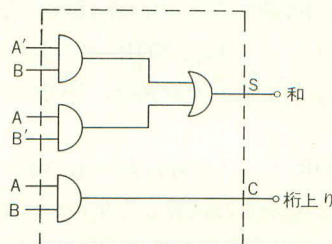



図 5-18　半加算器の論理図

2つの入力（A と B）および2つの出力（和 S と桁上り C）だけの回路を

作るには，他に，インバータ（図 5-19 参照）が2つ必要である。

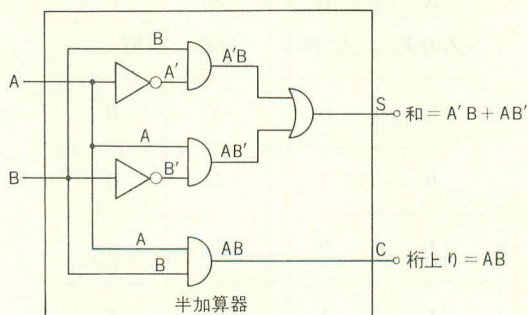



図 5-19 半加算器の論理図（否定入力がない形）

上の半加算器の論理回路は正しいけれども，さらにもっと簡単なもの（すなわち，もっと素子の少ないもの）ができる。和を表わすのに $S=(AB)'(A+B)$ の式から論理回路を作ると，もっと簡単なものが得られる。もちろん，真理値表，マップ，または式の展開などによって，この式は，$S=A'B+AB'$ と同じであることがわかる。桁上りの式 $C=AB$ を，和の式の一部からとり出すことができ，別に作らなくてもいいというところが注目すべき点である。

この回路を 図 5-20 に示す。

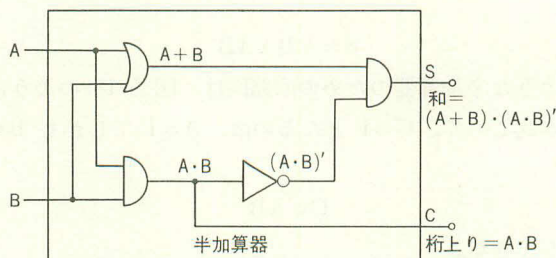



図 5-20 半加算器の簡単な回路（否定入力はない）

半加算器を表わすもう1つの方法は，NOR ゲートだけを使う方法であるが，これを 図 5-21 に示そう。このような回路は，NOR ゲートしか使わないシステムで用いられる。

このほか，Exclusive-OR ゲートと呼ばれるものがしばしば用いられる。これは，2 つの入力が，等しいかまたは等しくないかによって，異なる値を出すゲートである。すなわち，比較器回路であり，2 つの入力が等しくないときは，値は1となり，等しいときは0となるような働きをする。回路を作る前に，まず，真理値表を書いてみると，図 5-22 のようになる。

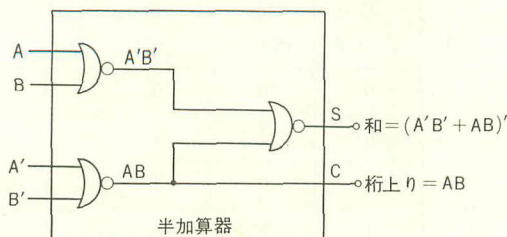



図 5-21 NOR ゲートだけを使った半加算器の論理図

A	B	出　力
0	0	0
0	1	1
1	0	1
1	1	0

図 5-22 比較器回路（Exclusive-OR）の真理値表

上の比較回路の式は，不一致出力$=A'B+AB'$となり，この回路を図 5-23に示す。この式および回路は，半加算器の一部分と同じものである。

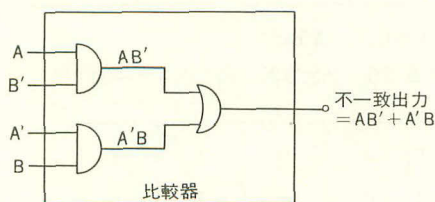



図 5-23 比較器（Exclusive-OR ゲート）の論理図

全加算器は，3 つの入力，すなわち，A, B および下の桁からの上がり（これを桁入りと呼ぶ）を持つ。以後，**桁上り**（carry out, C_o）と**桁入り**（carry in, C_i）とを区別して扱う。真理値表（図 5-24）を書いて，桁上りと桁入りに対する式を作ってみよう。和の式は次のようになる。

$$S=A'BC_i'+AB'C_i'+A'B'C_i+ABC_i$$

A （入力 1）	B （入力 2）	C_i （桁入り）	S （和）	C_o （桁上り）
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

図 5-24　全加算器の真理値表

次に，この和（S）の式を簡単にするためにカルノー図を用いてみる。真理値表からマップ（図 5-25）はただちに作ることができる。

図 5-25　全加算器の和のカルノー図

C_i \ AB	00	01	11	10
0			1	
1		1	1	1

AB, AC_i, BC_i

図 5-26　全加算器の桁上りに対するカルノー図

$C_o = AB + AC_i + BC_i$

マップから，この式は簡素化できないことは明らかで，せいぜい次のような式にするぐらいである。

$$S=C_i'(A'B+AB')+C_i(A'B'+AB)$$

桁上りについてのマップを 図5-26 に示す。1 が隣合っているから，簡素化は可能で，次のような式ができる。

$$C_o=AB+BC_i+AC_i=AB+C_i(A+B)$$

上の考察から，全加算器のための回路は 図 5-27 となることがわかる。

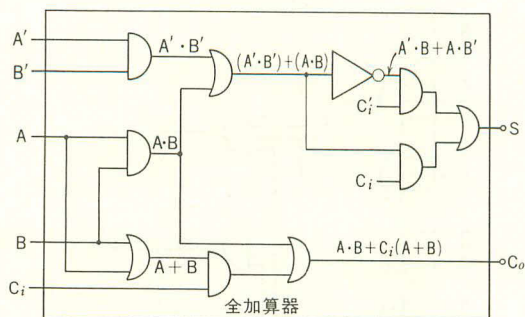



図 5-27　全加算器の論理図

この論理図は複雑に見えるかもしれないが，それは線が全部引かれていて交さしているからである。交さしている線のところに文字を書き込んで，別々にすれば，もっと理解が容易になる。同じものを得るのに，NAND ゲートまたは NOR ゲートを使うこともできる。NAND ゲートに適するように，式を変形してみよう。2 重否定とド・モルガンの定理を使うことにより，次のように変形できる。

$$\begin{aligned}S&=A'B'C_i+A'BC_i'+AB'C_i'+ABC_i\\&=[(A'B'C_i+A'BC_i'+AB'C_i'+ABC_i)']'\\&=[(A'B'C_i)'(A'BC_i')'(AB'C_i')'(ABC_i)']'\end{aligned}$$

これは NAND の形である。

$$\begin{aligned}C_o&=A'BC_i+AB'C_i+ABC_i'+ABC_i\\&=[(A'BC_i+AB'C_i+ABC_i'+ABC_i)']'\\&=[(A'BC_i)'(AB'C_i)'(ABC_i')'(ABC_i)']'\end{aligned}$$

これは NAND の形である。

マップを用いて簡単にした C_o の表現から NAND の形のものを導くこともできる。簡単にした式は，$C_o=C_i(A+B)+AB$ であるから，これを NAND

の形にすると，$C_o=[(AB)'(AC_i)'(BC_i)']'$ となる。これは前の形より NAND ゲートが1つ少なくてすむ。しかし，前の形にある $(ABC_i)'$ は和を作るときにも使われているので，項の数は同じである。和 (S) と桁上り (C_o) の両方のカルノー図を見ると，$(ABC_i)'$ というのは，それぞれのマップで，同じ場所に1のあるただ1つの項であることがわかる。全加算器の回路の例を図 5-28 に示す。NOR ゲートだけの全加算器は，自分で作ってみて欲しい。

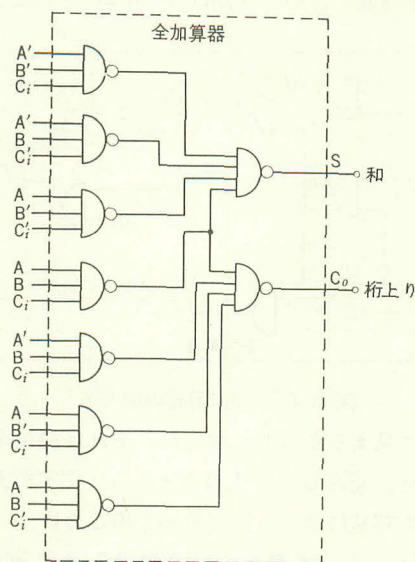



図 5-28 NAND ゲートによる全加算器の論理図

X (入力 1)	Y (入力 2)	D (差)	B (借り)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

図 5-29 半減算器 (X-Y) の真理値表

2進法の減算を行なうときは，最も下の桁については**半減算器**だけあれば求めることができる。上の桁については，**全減算器**が必要で，下の桁からの借りが入力として必要である。**差** (difference) および**借り** (borrow) を出力とす

る回路を設計してみよう。まず，真理値表を作る。次にマップを使うなどして，式を簡単にし，回路を作ってみる。X と Y を入力とする半減算器の真理値表を 図 5-29 に示す。

真理値表より，X=1 で Y=0 (Y′=1) か，または X=0 (X′=1) で，Y=1 ならば，差 D は1であることがわかる。式では，D=XY′+X′Y と書ける。借りについては，X=0 (X′=1) で Y=1 ならば借り B は1である。すなわち，B=X′Y である（差の式の中に，同じ項がある）。AND と OR および INVERTER のゲートを使って，半減算器 D=XY′+X′Y, B=X′Y を与える回路を組むと，図 5-30 のようになる。

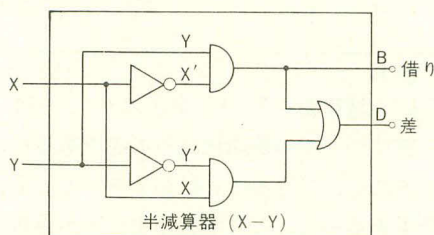



図 5-30 (X-Y) の半減算器の論理図

全減算器に必要な入力は，(X-Y) の XおよびYのほかに，下の桁への貸し (borrow in) とがある。この桁で出力となるものは差と，借りである。以後その借りを B_o (borrow out)，入力となる下の桁への貸しを B_i (borrow in) と記す。まず，全減算器の働きの真理値表（図 5-31）を作ることにしよう。

X (入力 1)	Y (入力 2)	B_i (貸し)	D (差)	B_o (借り)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

図 5-31 全減算器 [(X−Y)−B_i] の真理値表

真理値表から，差と借りの論理式を作る。

$$D = XY'B_i' + X'YB_i' + X'Y'B_i + XYB_i$$

$$B_o = X'YB_i' + X'Y'B_i + XYB_i + X'YB_i$$

これらに対するカルノー図（図 5-32）を書いて，簡素化を試みよう。

(a) 差 D

B_i ＼ X / Y	0 / 0	0 / 1	1 / 1	1 / 0
0		1		1
1	1		1	

(b) 借り B_o

B_i ＼ X / Y	0 / 0	0 / 1	1 / 1	1 / 0
0		1		
1	1	1	1	

図 5-32　全減算器［(X−Y), B_i］のカルノー図

このようにして，差についての簡素化は不可能であるが，借りについては，集め直して簡単にすることができることがわかる。しかし，2 つのマップには共通な部分が 3 個所もあるので，これを利用するのが得策である。ここでは，NOR 回路（図 5-33）による全減算器を示すから，NAND 回路によるものを

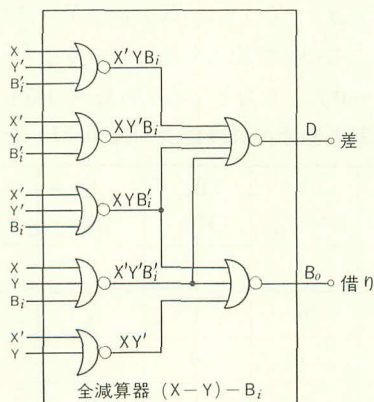



図 5-33　NOR ゲートだけによる全減算器〔(X−Y)−B_i〕の論理図

各自で作ってみるとよい。表現を簡単にするため，否定項（X′, Y′ など）を求めるのに必要なインバータを書いてないが，NOR ゲートが 7 個使われている。この回路のための式を導こう。まず，マップで 1 のない項をとり出してきて OR で結ぶと，これは D′ の式になる。これを否定すれば，D を NOR で表わしたものになる。

$$D' = X'Y'B_i' + XYB_i' + X'YB_i + XY'B_i$$
$$D = (X'Y'B_i' + XYB_i' + X'YB_i + XY'B_i)'$$

同様に，借り B_0 については，次のようになる。

$$B_o' = X'Y'B_i' + XYB_i' + XY'$$
$$B_o = (X'Y'B_i' + XYB_i' + XY')'$$

要　約

ブール代数を導入して，実際の問題に応用した。そして，AND, OR, INVERTER, NOR および NAND 関数について考察した。論理図を“読み”取ったり，ブール代数式から論理図を作れることは大切なことである。演算の規則を応用して，簡単な形にしたり，回路に適した形に作り直すことができる。ブール代数における非常に重要な定理の1つに，ド・モルガンの定理があるが，これは次のようなものである。

$$(X+Y)' = X' \cdot Y'$$
$$(X \cdot Y)' = X' + Y'$$

また，ブール代数式の変形や演算を容易にするために，カルノー図が使われる。

以上の事柄を結びつけて理解をしっかりさせるために，非常に普遍的な（そして重要な）回路をいくつか設計した。この中には半加算器，全加算器，半減算器，全減算器などがある。

問　題

1. $(A+A'B)(A+B')$ を簡単にしなさい。
2. 次の式の論理図を作りなさい。
   $AB' + (AB) \cdot (A+B)$
3. 次の論理式を簡単にしなさい。
   $UV + VW + UW + VW'$
4. 次の式を簡単になさい。
   $A'BC + ABC' + ABC + A'B$
5. NOR 論理回路だけで次の式を表わしなさい。
   $(AB' + A'B)C$
6. NAND 回路だけで次の式を作り上げなさい。
   $(A'B' + AB)C + (A'B + C')D$

7. 次の式のカルノー図を書きなさい。

ABC+BC′+AC+BC

8. 次の式のカルノー図を書いて，簡単な式に直しなさい。

XY+Z+XYZ+Z′Y

9. 次のカルノー図から論理式を作りなさい。

U V ＼ X / Y	0 / 0	0 / 1	1 / 1	1 / 0
0 0	1			1
0 1	1			1
1 1				
1 0	1			1

10. NAND ゲートだけで全減算器の論理図を作りなさい。

第II部　計算機回路

0110＝6

計算機の論理ゲート

電子計算機の回路の分野は，非常な速さで発展し，絶えず変化している。初期には，計算機の回路（論理回路）はリレーやスイッチを使って巧みに作られていた。最初の計算機はわずかしか記憶装置を持っていないか，またはまったく持っていないで，配線によって作られた，すなわち固定したプログラムだけで動作していた。その後，記憶装置という特徴がつけ加えられて，プログラムを記憶することが可能になり，電子計算機の利用に関する観念はまったく変化した。また，リレーは遅くて大きくそして機械的信頼度が低いため，計算機回路の主流からやがてはずれることになった。真空管（3極管と2極管）は短期間使われたが，それらもすぐにその大きさや使用電力（数千本使っている電子計算機ではヒーターの電力だけでもかなりのものである）およびその信頼性のため，使われなくなった。現在では，固体素子が計算機の分野にとり入れられ，完全に主導権をとることになった。スイッチングトランジスタ，スイッチングダイオード，トンネルダイオード，電界効果トランジスタ，ツェナーダイオード，シリコン制御整流器（SCR），ユニジャンクショントランジスタ，パワートランジスタなどは，すべて固体素子の長所を持っている。すなわち，形が小さく，ヒーター電力が必要なく（そして予熱時間が必要ない）そして高い信頼性を持っている。初期には，これらの素子は，軽くて小さく，消費電力が少ないといった点から，海上や宇宙で使用するのに都合よく，軍事の関係で使われていた。一度使われ始めると，最初の費用が高いという問題は克服され，その高い信頼性のために，民間で広く使われるようになった。図 6–1a は，実際に電子計算機回路に使われているプリント回路板を示したものである。この章ではこれらの回路について検討を加える。

過去 2,3 年の間に，固体電子工学技術の進歩によって，再び計算機回路の性

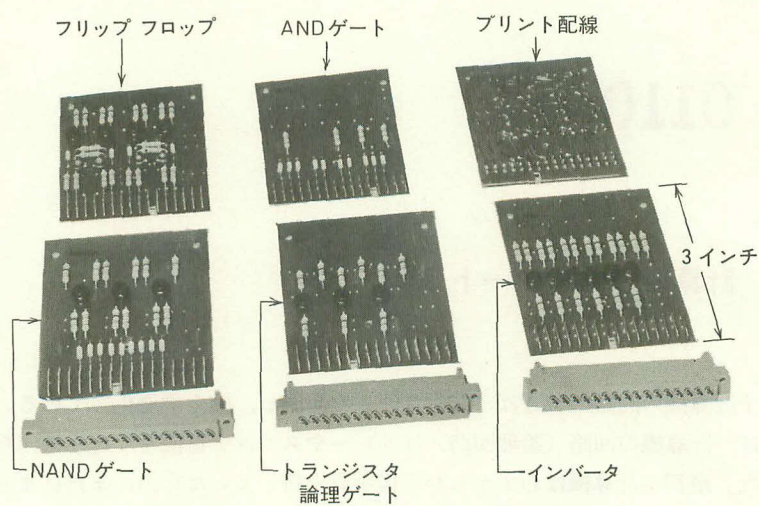



図 6-1a プリント回路板の電子計算機回路 (Digital Electronics, Westbury, N.Y. の好意による)。

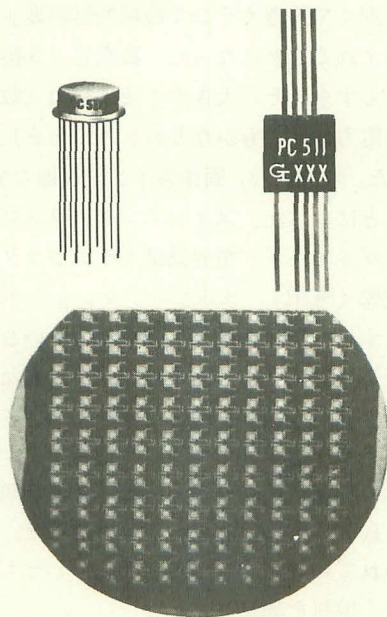



図 6-1b マイクロ エレクトロニクス(集積)素子から成る電子計算機回路

質はかわった。1 組あるいは数組のゲートの入った集積回路が 1 つの製作過程で作られるようになった。これらの新しい素子 (component) (図 6-1b 参照) は一般に マイクロ エレクトロニクス と呼ばれている。全体として，大きさは 2,3 桁小さくなり，非常に高い信頼度を持っている。現在のところ大規模に使うにはコストが障害となっているが，急速に安くなりつつあり，新しい計算機は，マイクロ エレクトロ ニクス回路を使うようになっている。現在，いくつかの異なった型のマイクロ エレクトロ ニクス回路がある。そして，よりよい回路を開発するための研究の結果，回路の構成と用いる基本的な要素は絶えず進歩している。集積回路というのは，薄板上にシリコンを拡散させ，腐食させてすべての素子をつなぎ合わせたものである。これはトランジスタを作るのと似た製法である。外部的に加工することは最小限にしてあり，大部分の処理は自動的に行なわれるので，信頼度は非常に高い。しかしながら，値段がまだ高いので回路技術面での改良が検討されている。ハイブリッド回路は，別々に作られた部分品を使って，1 つの回路を作るが，しかしこれらはすべて同じ大きさの小型集積パッケージにまとめられる。

計算機回路の分野の絶えまない発展にもかかわらず，非常によく使われている基本回路として少なくとも 6 種類ある。そして，それらはそれぞれ非常によく標準化されている。驚くにはあたらないが，回路は時代とともにその複雑さを増しているので，より単純な回路から議論を始めるのは，また歴史的発展をとらえることでもある。半導体素子を使った装置は非常によく知られており実際に見かけることも多いので，典型的な構成回路を述べるのにそれを用いる。計算機の OR ゲートと AND ゲートは，ダイオードと抵抗の組合せで作られ，ゲートすなわち論理要素にはダイオードが使われる。インバータ回路は信号を逆にする――すなわち否定する――ために，**能動素子** (active device)*，この場合トランジスタを必要とする。また NAND ゲートまたは NOR ゲートは，ダイオード，抵抗およびトランジスタの組合せで作られる。ダイオードは論理要素として，またトランジスタは否定をとるために使われている。

6-1 ダイオードによる AND ゲートと OR ゲート

ダイオードゲートは 1 つの抵抗と，入力に見合うだけの**ダイオード** (diode)

*〔訳注〕 能動素子とはその等価回路に増幅信号源を含むような回路素子である。

によって作られる*。AND 機能として普通使われているのは，図 6-2 の回路である。

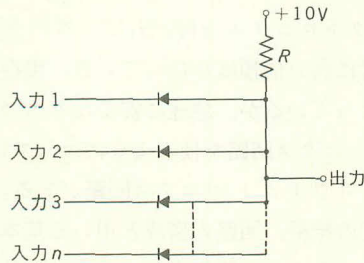



図 6-2　ダイオード AND ゲート

この回路で重要な2つの電圧値は 0V（ボルト）と +10V である。この2つの電圧は論理状態0と1を表わすために使うことができる。0V を論理状態0に，そして +10V を論理状態1と仮定する。もしすべての入力が 0V (0) なら,出力は 0V(0) である。もしどれかのダイオードの入力が 0V であると，出力は 0V である。このことは，ダイオードゲートの働きを理解していれば，明らかなことである。どんなダイオードも，順方向にバイアスされていれば，電圧降下は非常に小さい（10 分の 2,3 ボルト）。この電圧は 10V とはかなり違うので，容易に区別がつけられる。理想的にみて（または便宜上），ダイオードの電圧降下は0と考えることができる。今後，出力が 0V であるというときには，近似的にダイオードの電圧降下が0であるとみなせることを意味する。

ここで，ダイオードの特性について考えてみよう。図 6-3 に示したものは，順方向にバイアス**した状態での典型的なダイオード特性である。この図からわかるように，ダイオードを通る電流が大きく変化しても，ダイオードの両端の電圧は，あまりかわらない。これは，ダイオードが順方向（陽極から

*〔訳注〕ダイオードは，検波器や整流器として広く使われている半導体素子で，電流を一方にだけ流す働きを持つ。これは，N 型半導体（電子が自由に動くことのできる半導体）と，P 型半導体（ホール，すなわち，電子を吸いとられて正に帯電しているものが自由に動くことのできる半導体）を接合したものである。

これを表わすのに，上の記号を用いるが，このように書いたときは，電流は右から左への方向だけ流れ，その逆のときは絶縁体と同じになる。

**〔訳注〕順方向にバイアスするとは図のようにダイオードの両端に電圧がかかることであり，図 6-3 では縦軸から右の領域である。逆方向にバイアスするということは，図 6-2 とは逆に電圧がかかることである。

陰極への通常の電流の方向）には低い抵抗を持っていることを意味する。逆方向にはわずかに電流が流れ，電流の微少の変化に対して，大きく電圧が変化する。これは，ダイオードが逆方向には高い抵抗を持っていることを意味する。ダイオードを順方向にバイアスするためには，陽極の電圧は，陰極の電圧より高くしておかなければならない。

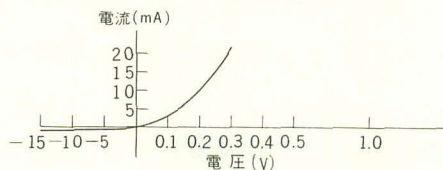



図 6-3　ダイオード特性

1入力ダイオードゲートでは論理演算はできないが，それでも回路の電気的働きを説明するのには使える（図 6-4a 参照)。入力が 0V(0) のとき，+10V の供給電圧をかけると，ダイオードの陽極側に接続された抵抗を通して，ダイオードを順方向にバイアスさせる。ダイオードが順方向にバイアスされると，ダイオードの両端には 10 分の数ボルトの電位差があるに過ぎない。ダイオードを通る電流は $10/R$ (mA) (R は kΩ) 以下に制限される。陽極とアース間の出力電圧は約 0V である。

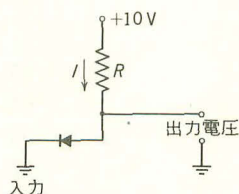



図 6-4a　入力 0V におけるダイオードゲートの出力

入力電圧が 10V であるときには，ダイオードにも抵抗にも電流が流れず，抵抗の両端に電位差は生じない（図 6-4b 参照)。出力電圧（陽極とアースと

図 6-4b　入力 +10V のダイオードゲート

の間の電圧）は +10V で，入力電圧と同じである。

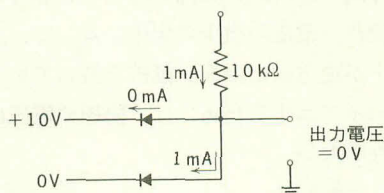
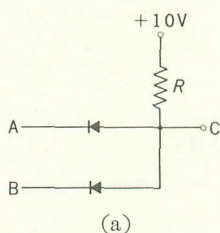



図 6-5　入力 0V と +10V のダイオード AND ゲート

論理演算を行なうことのできるゲート，すなわち，2つ以上の入力のあるダイオード ゲート を考えてみよう（図6-5参照)。もし，どちらかがアース電位で他が +10V なら，アースされている方のダイオードは順方向にバイアスされたままである。そのダイオードの両端の電位差は 0V なので，出力は 0V である。そして，電流はすべて導通している（順方向にバイアスされた）ダイオードを流れる。もし2つの入力が，両方とも 10V なら，出力は 10V になる。2つのダイオードは，逆方向にバイアスされ，それらのダイオードには電流は流れない。電圧の組合せは電圧真理値表によって簡単に表わせる。上に考えたダイオード ゲート についての表を 図6-6 に示す。

A（入力 1）	B（入力 2）	C（出力）
0V	0V	0V
0V	+10V	0V
+10V	0V	0V
+10V	+10V	+10V

(b)

図 6-6 （a）ダイオードゲートと （b）電圧真理値表

すべての電圧は理想値であって，順方向にバイアスされたダイオードの電圧降下は 0，そして逆方向の電流は0としている。

A（入力 1）	B（入力 2）	C（出力）
0	0	0
0	1	0
1	0	0
1	1	1

0V≡0
+10V≡1

図 6-7　AND ゲートの論理真理値表（正の論理）

論理値 0 と 1 に対し，それぞれ 0V と +10V を対応させると，論理真理値表を 図 6-7 のように作ることができる。

この関数では，A と B が 1 のときだけ 1 の出力がある。論理式で表わすと，C=AB である。これは AND 関数であり，このゲートは AND ゲートである。出力は両方の入力があるときにだけ得られる。

このように解釈するのは非常に一般的ではあるが，+10V を論理の 1，0V を論理の 0 としたのは勝手に選んだものである。この対応を逆にすると，電圧真理値表から異なった論理真理値表（図 6-8）が得られる。

A（入力 1）	B（入力 2）	C（出力）	
1	1	1	
1	0	1	0V≡1
0	1	1	+10V≡0
0	0	0	

図 6-8　ダイオード OR ゲートの論理真理値表（負の論理）

出力関数は，両方の入力が 1 のとき，A が 1 で B が 0，または A が 0 で B が 1 のとき，値 1 を持つ。方程式で表わすと，

$$
\begin{aligned}
C &= AB + AB' + A'B \\
&= A(B+B') + A'B \\
&= A \cdot 1 + A'B \\
&= A + A'B \\
&= A + B \quad \text{（5 章で証明した恒等式を使う）。}
\end{aligned}
$$

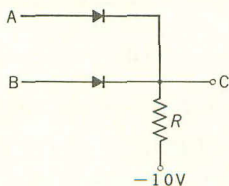



図 6-9　ダイオード OR ゲート

この第 2 の論理水準の定義では，関数は OR になった。そこで，このゲートは OR ゲートと考えられる。2 つの定義を区別するために，最初のものを，**正の論理**と呼ぶ。なぜなら，論理の 1 が高い方の電圧（0V に対して +10V）に選ばれているからである。2 番目の論理の選択は，**負の論理**と呼ぶ。すなわち，論理の 1 が，より低い電圧（+10V に対して 0V）に選んであるからであ

る。いままで論じてきたゲートは，正の論理の AND ゲートである。

ダイオードを要素に使い，0V と +10V を論理水準としている一般的なもう1つのゲートを 図6-9 に示す。

このゲートは，入力がダイオードの陽極側に与えられ，抵抗が −10V に接続されているという点で，先のとは異なっている。もし，2 つの入力が 0V なら，出力もほぼ 0V である（順方向にバイアスされたダイオードであるから，10 分の 2,3 ボルトの電圧降下）。一方に 0V，他方に +10V の入力があれば，順方向にバイアスされたダイオードにより，出力は +10V（図 6-10）となる。入力が 0V の方のダイオードには，逆方向のバイアスがかかり，電流は流れない。

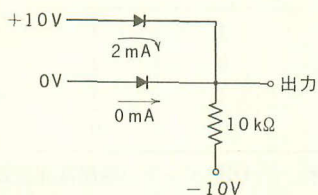
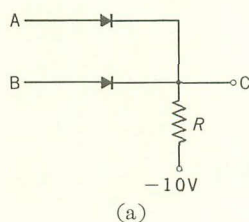



図 6-10　ダイオード OR ゲート（入力 +10V と 0V）

A（入力 1）	B（入力 2）	C（出力）
0	0	0
0	+10	+10
+10	0	+10
+10	+10	+10

(b)

図 6-11　(a) ダイオード OR ゲートと (b) 電圧真理値表

両方のダイオードに +10V の入力があれば，出力もまた +10V である。この回路で，0V と +10V の入力に対し，出力を理想的な 0V と +10V としたときの電圧真理値表を 図6-11 に示してある。

0V を論理の 0，+10V を1と定義すれば，対応する正の論理真理値表（図 6-12）が得られ，ゲートは OR ゲートであることがわかる。論理の1が高い方の電圧であったから，ゲートは正の論理の OR ゲートである。すぐ想像がつくことだが，負の論理の論理真理値表（図 6-13）を作れば，これは AND 関数を表わしている。正の論理演算の2つのゲートをまとめて 図6-14 に示しておく。

A (入力 1)	B (入力 2)	C (出力)	
0	0	0	0V≡0
0	1	1	+10V≡1
1	0	1	
1	1	1	

図 6-12　ダイオード OR ゲートの論理真理値表（正の論理）

A (入力 1)	B (入力 2)	C (出力)	
1	1	1	0V≡1
1	0	0	+10V≡0
0	1	0	
0	0	0	

図 6-13　ダイオードによる AND ゲートの論理真理値表（負の論理）

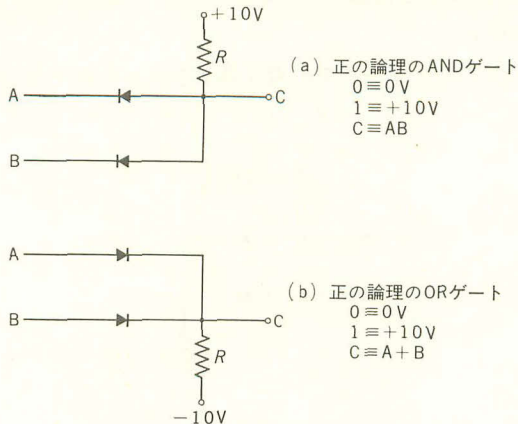



図 6-14　正の論理のダイオードゲート

問題 6-1　負の論理の OR ゲートの電圧真理値表と論理真理値表を書きなさい。

問題 6-2　図 6-15 の回路で，適当な論理の電圧を示し，電圧真理値表を書き入れなさい。

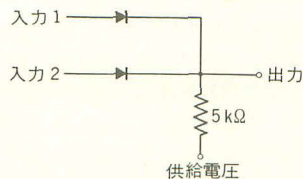



図 6-15　問題 6-2 のダイオード回路

多段論理接続における負荷　AND と OR ゲートを使って，論理回路を組み立てるとき，接続することのできる段数は制限される。この制約は演算速度と電圧レベルの面から生じる。正の論理ゲート（図 6-14）の回路について，その制約が出てくる理由を説明する。AND ゲートと OR ゲートが接続されたときを考えてみよう（図 6-16）。

入力 A または B が 0（0V）なら，点Cの電圧は $+V_D$，すなわち，順方向にバイアスのかかったダイオードの両端の電圧降下である。もし D が 1(+10V) なら，E での出力は $+10-V_D$ すなわち約 +10V である。これらの入力については，ゲートは正しく動作する。さて，A と B が両方とも 1 の場合の働きを考えよう。AND ゲートの 2 つのダイオードは逆方向にバイアスされ，AND ゲートだけなら，C 点の電圧は +10V になるべきである。ところが，開いた回路*に接続されていないため，そうはならない。OR ゲートは有限の入力インピーダンス**を持っていて，それが負荷となってC点の電圧を下げてしまう。カット オフ*** されている入力のAとBのダイオードをとり去って考えるとはっきりする（図 6-17）。

図の OR ゲートのダイオードでの電圧降下を 0 とみなすと，点C と E の電

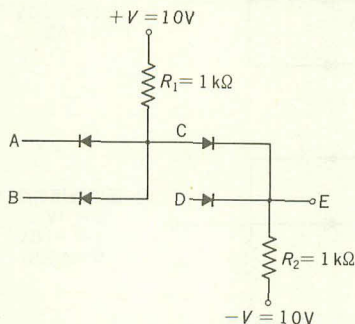



図 6-16　ダイオードによる AND-OR ゲート

---

*〔訳注〕開いた回路とはその回路への入力インピーダンスが無限大ということである。

**〔訳注〕増幅器にある信号を入力するとき，その信号は，増幅器の中で，エネルギーを吸いとられることになる。これは，入力端子間の抵抗（入力インピーダンス）の大きさで決まる。一般に真空管増幅器は入力インピーダンスが大きいので，入力すべき信号源の負荷を考えなくともよいが，トランジスタ増幅器では，トランジスタの内部抵抗が小さいため，入力インピーダンスは重要なものとなってくる。

***〔訳注〕カット オフ（cut off）とはダイオードが逆バイアスとなって電流が流れないこと。

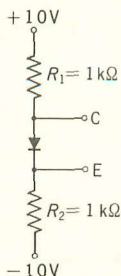



図 6-17　AND-OR ゲートの回路の一部分

圧は抵抗 R_1 と R_2 の電圧分割によって決定される。この例では，2 つの抵抗はともに 1kΩ であるから，点CとEの電圧を計算すると，

$$\frac{R_2}{R_1+R_2}(20)-10=0\ (\mathrm{V})$$

となる。この 0V は論理の 0 である。ところが，AND ゲートの 2 つの入力が 1 であれば，出力は 1(+10V) でなくてはならない。この場合，明らかにゲートは正しく動作していない。R_2 を R_1 より大きくすれば，この点は改善することができる。R_2 を R_1 の 10 倍にすれば，C と E での電圧は

$$\frac{10R_1}{R_1+10R_1}(20)-10=\frac{10}{11}(20)-10=18.2-10=8.2\ (\mathrm{V})$$

である。これは多分ゲートが正しく動作するために十分な電圧であろう。しかし，もし AND ゲートが 2 つの OR ゲートに接続されたらどうなるだろうか。2 つの OR ゲートの負荷は 2 つの抵抗 R_2 が並列に接続されたものになる。そのような場合，$R_2=\frac{1}{2}(10R_1)$ であるから，出力は

$$\frac{5R_1}{R_1+5R_1}(20)-10=\frac{5}{6}(20)-10=16.6-10=6.6\ (\mathrm{V})$$

となる。この電圧では，正しく動作するのに十分でないだろう。R_2 をもっと大きくとることも可能である。しかし動作速度の制約があるから，R_2 をいくらでも大きくすることはできない。ゲートの出力側には，配線とか，ダイオードなどにより，わずかではあるが漂遊容量 (C) がある。ゲートを単純に図 6-18 のように表わして，R_2 の効果を見てみよう。

2 つの入力が低く (0V) なると，出力もまた 0V になるはずである。しかし，入力が 0 になっても，OR ゲートのダイオードは，コンデンサに蓄えられている電荷のため，逆方向にバイアスされる。そしてその電荷が R_2 を通って

放電されるに従い出力は 0 になる。このように，時定数* R_2C は出力が 0V になる速さを左右する。

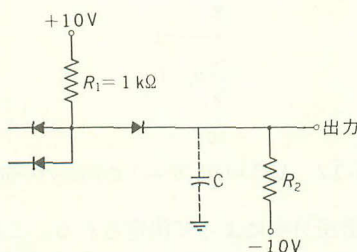



図 6-18　漂遊容量を含めたダイオード AND-OR ゲート

C を 100 $\mu\mu$F, R_2 を 100 kΩ とすると，時定数は

$$R_2C=(100\times10^{-12})(100\times10^{3})=10\times10^{-6}=10\ (\text{マイクロ秒})$$

もし十分低い電圧**になるまでに時定数の 4 倍かかるとすれば，ゲートは 40 マイクロ秒（25 k Hz***）より速く動作することはできない。いま選んだ値は特に大きいが，R_2 に限界があることは明らかである。

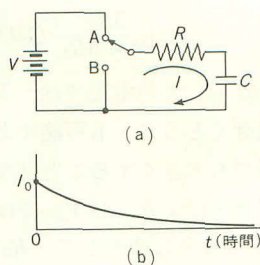
前の回路を見れば，AND ゲートの負荷を軽減する別の方法があることがわかる。すなわち，もし R_1 を R_2 より小さくすれば，出力は高くできる。しかし，この方法もまた欠点がある。R_1=1kΩ での入力ダイオードを流れる電流を考えてみよう。最大の値は 10V/1kΩ，すなわち 10mA よりわずかに少ない程度である。もし，出力電圧レベルを改善するために，R_1 を 100 Ω にすると，ダイオードには約 100mA 流さなくてはならなくなる。明らかに，R_1 を極端に

*〔訳注〕 R と C による図 (a) のような回路で，スイッチが A にあったとする。このとき，B にたおしたときを時刻 0 とすると，図 (b) のような電流の変化が見られる。このとき，この曲線は

$$I=(1-e^{-\frac{t}{RC}})\cdot I_0$$

となる（I_0 は時刻 0 のときの電流）。I/I_0 の比が $1-e^{-1}\approx0.37$ になる時間，すなわち RC の値を時定数という。

**〔訳注〕 98%

***〔訳注〕 Hz は周波の単位 (cycle per second) であって，H. R. Hertz にちなんで国際的に用いられるようになった。

減らせば論理ダイオードの選択に過度の制約がかかることになる。ダイオードや回路は計算機の中で非常に多く使われているから，できるだけ少ない電力で働くことが望ましい。結論として，AND ゲートと OR ゲートを用いる場合，次々に接続する論理ゲートの数には，制約があることがわかった。この問題に対する1つの解決法は，過負荷を防ぐためにインバータを使うことである。インバータは，基本的には，増幅器であり，信号を作り直し，負荷が正しくかかるようにする。インバータの動作を考える前にダイオード回路の問題をためそう。

問題 6-3　図 6-19 の回路において，次の場合の実際の出力はどうなるか。

1.	2.	3.	4.
A=1	A=1	A=1	A=1
B=0	B=0	B=1	B=1
D=1	D=0	D=0	D=1

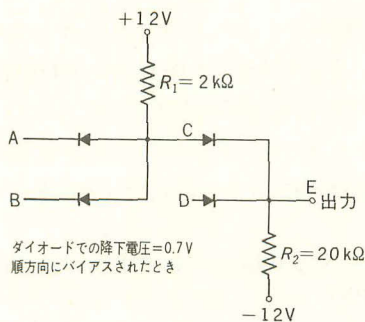



図 6-19　問題 6-3 の回路

6-2 トランジスタ インバータ ゲート

反転の作用には，真空管やトランジスタのような能動素子が必要である。真空管のグリッドに信号を与えると，増幅され，反転された信号が得られる。ト

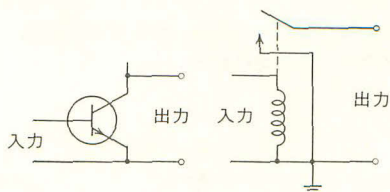



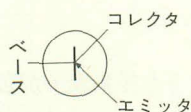
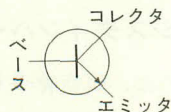
図 6-20　トランジスタ スイッチ

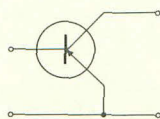
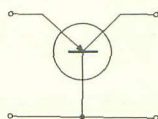
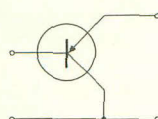
ランジスタ* (transistr) では，ベースとエミッタの間に信号が与えられると，増幅され，反転された信号がコレクタとエミッタの間に出てくる（エミッタ接地回路)。計算機の回路では，トランジスタ の主要な機能は論理的な反転であって，増幅作用はゲート間の負荷を適正にするために用いられる。

トランジスタの性質を，スイッチング回路に重点をおいて，概観してみよう。論理の目的で使用するためには，トランジスタは0または1という状態を，はっきり区別がつくように表わせなくてはならない。考えられる最良の状態は，トランジスタが完全に OFF または完全に ON になっているときの状態である。これらの状態のもとでの働きを，理想的な素子としてのスイッチの働きと比較すると，トランジスタがスイッチング回路でいかにうまく動作しているかがわかる。

図 6-20 にトランジスタ スイッチ と リレー スイッチの比較を示してある。入力が ON になると，リレーは接点を閉じ，出力は接地電圧になる。しかし，トランジスタスイッチでは本当の接地電圧にはならない。これはトランジスタが完全に ON になったときでも，コレクター-エミッタ 間に電圧降下が生じるからである。リレーの接点が開かれると，完全な開回路になるが，トランジスタ

---

*〔訳注〕 P 型半導体と N 型半導体を，P-N-P，または N-P-N の順に，サンドイッチ型に接合すると，PNP トランジスタまたは NPN トランジスタになり，三極真空管と同じように，増幅作用を持つことができる。PNP 型は (a)，NPN 型は (b) のように書き，電極には，次に示すような名前がついている。

(a) PNP 型

(b) NPN 型

エミッタを接地した回路をエミッタ接地，コレクタを接地した回路をコレクタ接地，ベースを接地した回路をベース接地回路という。

エミッタ接地

ベース接地

コレクタ接地

では，カットオフの状態でも，微少な電流が流れ，やはり理想的な状態とはいくぶん異なっている。しかしながら，このような問題があったとしても，トランジスタの利点は，その非常に速いスイッチング時間（リレーのミリ秒に対しマイクロ秒以下）と寿命の長さ（リレーの数十万回に較べて，何百万回の動作に耐える）にある。

エミッタ接地回路の特性を調べれば，理想の状態からの ずれ やスイッチング素子としての基本的な働きがよりはっきりと判るようになるだろう。典型的な特性*を 図6-21 に示す。

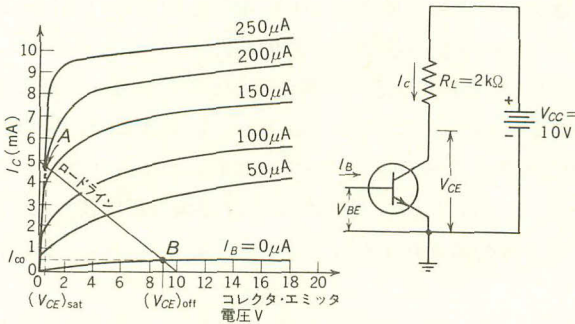



図 6-21　エミッタ接地回路の特性

トランジスタが動作する領域を決定するために，負荷抵抗をつけて考える。完全に ON のとき，トランジスタの電流は，ある一定の値すなわち飽和値まで増加する。この状態におけるトランジスタでの電圧降下は飽和電圧 (saturated voltage) と呼ばれ，それを $(V_{CE})_{sat}$ と書く。

---

*〔訳注〕 図の特性曲線は，エミッタ接地としたトランジスタの，ベースとコレクタの電流の関係を示したものである。これは，横軸にコレクタとエミッタ間の電圧をとり，縦軸にコレクタ-エミッタ間に流れる電流を示したもので，この曲線は，ベース・エミッタ間を流れる電流によって異なった曲線となる。もし図 6-21 のように，負荷抵抗を一定にすると，トランジスタの動作は，特性曲線のロードラインという直線上を動くことになる。すなわち，トランジスタの内部抵抗がいくらであろうとも，この両端の（コレクタ-エミッタ間の）電圧-電流の関係は，次式で与えられる。

$$I_C=(V_{CC}-V_{CE})/R_L$$

これは，図のような直線であり，負荷が一定である限り，ベース電流が決まれば，そのときの V_{CE} と I_C は一義的に決まる。I_B が $0\mu A$ ならば，点線で示したような電流，電圧となる。

ロードライン (load line) とトランジスタの特性曲線 (characteristic line) との交点 A は，トランジスタが完全に ON (または飽和) のときの動作点である。曲線から，出力の飽和電圧は，10 分の数ボルトであることがわかる。この 10 分の数ボルトが，スイッチに使うときの理想値，0 との ずれ である。これは，まだ OFF の値とは，十分区別がつけられる値 (図 6-21) である。負荷(曲)線と OFF 状態 ($I_B=0$) のトランジスタ特性曲線との交点Bは，供給電圧の 10 V よりいくらか低い値を示す。この電圧降下は，逆方向のバイアスのもとでの，少数のキャリア (minority carriers) の流れによるものである。カットオフ電流 I_{co} により，負荷抵抗での電圧降下 ($I_{co}R_L$) が生じ，理想的なスイッチ状態のもとでの値より OFF 電圧を低いものとしてしまう。カットオフ電圧と，飽和電圧の差ははっきり区別がつけられるので，一般に ON 電圧を 0V, OFF 電圧を供給電圧と考えることができる。正確な値が必要なときには,実際の値 $(V_{CE})_{sat}$ または $(V_{CE})_{off}$ を使えばよい。理想値を使うときは，単に論理的機能を述べていて，わずかな電圧の相違はたいした意味を持たない場合である。しかしながら,理想的な値は理論でだけ使われるのであって，実際の回路では，理想値からずれているということは，はっきり認めておくべきである。

スイチッングに使う場合，トランジスタは特性曲線の直線領域を通ってその状態をかえる。この領域を通過する時間が非常に短いので，スイッチングトランジスタは，飽和または カットオフ 領域のいずれかで動作するといわれる。トランジスタのスイッチング時間の分析は非常に複雑なので，ここでは触れないことにする。大部分の優れた スイッチング トランジスタ ではスイッチング時間が 1 マイクロ秒ないし数ナノ秒の範囲にあるということだけ，心にとめておけばよい。スイッチング時間を考慮に入れたインバータの設計の詳細について扱っている本が数多く出されている。この本での基本的な説明だけでは十分でないときには，他の問題でもそうだが，この問題についても必要に応じて他の参考書を調べるのがよい。

簡単なインバータ回路を，図 6-22 に示しておく。入力抵抗 R_B は，最小の入力電圧*でベース電流の値がトランジスタを飽和させるのに十分であるように選ばれる (コレクタ電流が飽和するように)。コレクタ抵抗 R_C は，あまり高くすると，I_{co} が流れたとき (OFF 状態)，出力電圧にかなりの電圧降下を

*〔訳注〕 正の論理で 1 に対応する電圧の最小値。

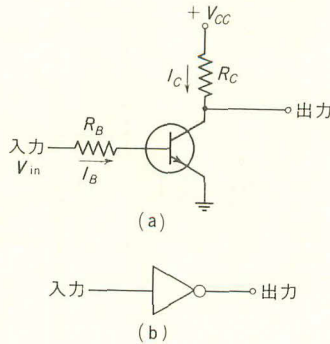



図 6-22 (a) インバータ回路 (b) 論理図

生じるので，高くできない。電圧真理値表と論理真理値表は，すぐ書けるだろうが，一応 図6-23 に示しておく。

入力が 0V のとき，トランジスタを ON にする電圧はなく，少数キャリアによる電流 I_{co} が流れる。出力電圧は $V_{CC}-I_{co}R$ である。I_{co} は数 nA(ナノアンペア, 10^{-9} A) ないし数 μA (マイクロアンペア, 10^{-6} A) で，R_C は数 kΩ なので，電圧降下 $I_{co}R_C$ は普通無視できる。+10V の入力では，トランジスタは ON にされる。完全に ON のとき，順方向にバイアスがかかり，ベース-エミッタ間の電圧は約 0.5V となる。その電圧は，入力電流が大きく変化しても，わずかしかかわらない。入力駆動電流 I_B は $(V_{in}-0.5)/R_B$ で表わされるから，R_B を 10kΩ, V_{in} を 10V とすると，$(10-0.5)/10(10^3)=0.95$ (mA) である。一般に，ON 状態のベース・エミッタ電圧を $(V_{BE})_{on}$ と書く。ON の状態での出力電圧は，前に述べたように，$(V_{CE})_{sat}$ であり，コレクタ電流は $I_C=(V_{CC}-(V_{CE})_{sat})/R_C$ である。代表的な値として，$R_C=2k\Omega$, $V_{CC}=10V$, $(V_{CE})_{sat}=0.3V$ とすると，$I_C=(10-0.3)/2=4.85$ (mA) である。

A (入力)	A′ (出力)
0V	+10V
+10V	0V

(a) 電圧真理値表

A (入力)	A′ (出力)	
0	1	0V≡0
1	0	+10V≡1

(b) 正の論理の真理値表

A (入力)	A′ (出力)	
1	0	0V≡1
0	1	+10V≡0

(c) 負の論理の真理値表

図 6-23 インバータの電圧真理値表と論理真理値表

問題 6-4 図 6-24 の回路の I_C と I_B を計算しなさい。

問題 6-5 図 6-24 で供給電圧 V_{CC} を +12V, R_B を 20kΩ, Vin を +12V とかえたときの I_C と I_B を計算しなさい。

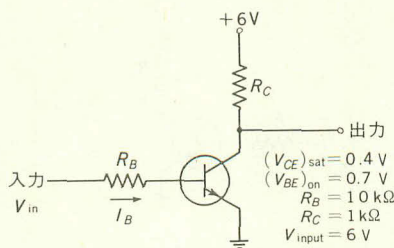



図 6-24　問題 6-4 と 6-5 のインバータ回路

6-3　トランジスタによる NAND ゲートと NOR ゲート

ダイオードの OR および AND ゲートと，トランジスタのインバータとを別々に考えてきたが，両者を1つのゲートに組み合わせると都合がよい。事実，AND ゲートや OR ゲートだけを接続して使うと，負荷の影響が起こってうまくない。1 つのゲートにつなげるゲートの数も，実際には，まったく制限されてしまう。負荷の整合と増幅のために，ダイオード ゲート 2,3 個ごとに トランジスタ インバータ が必要である。NAND ゲート（または NOR ゲート）は，1 つのブロックですべての論理機能を果たすことができるように開発されたものであった。

基本的なゲートは，論理ゲートのあとにインバータを加えて，作ることができる（図 6-25）。この図に示した論理ゲートは正の論理の AND ゲートであるから，全体の機能は AND-INVERSION すなわち NAND 機能を持つ。負の論理では，同じゲートは NOR 機能を持つ。このゲートの第 2 の型は，正の論理の OR ゲートのあとに PNP* トランジスタ インバータ をつなぐことによって作られる。この回路を 図 6-26 に示してある。NPN* 論理インバータ ゲート の具体的な回路とその電圧および論理真理値表を作ると，A′B′+A′B+AB′ のとき出力は 1 である：すなわち，

$$\begin{aligned}C&=A'B'+A'B+AB'\\&=A'(B'+B)+AB'\\&=A'+AB'\\&=A'+B'\end{aligned}$$

*〔訳注〕 P. 116 の脚注参照。

ド・モルガンの定理により，

$$A'+B'=(AB)'$$

$$C=(AB)' \qquad \text{NAND 関数}$$

負の論理では，出力の式は単に，

$$C=A'B'$$

これは，ド・モルガンの定理により，

$$C=(A+B)' \qquad \text{NOR 関数}$$

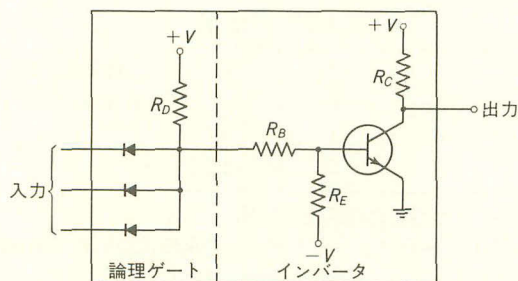



図 6-25　論理インバータ ゲート (NPN)

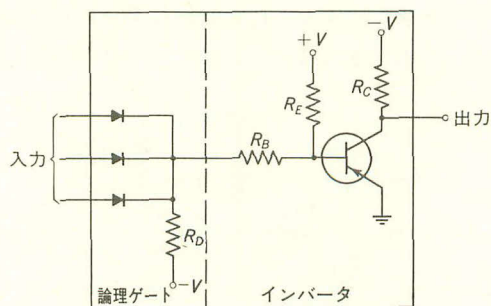



図 6-26　論理インバータ ゲート (PNP)

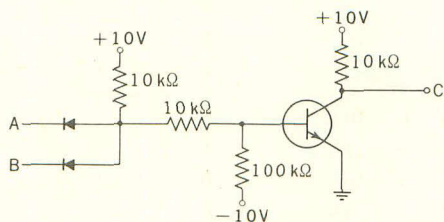



図 6-27　具体的な NPN 論理インバータ ゲート

A (入力 1)	B (入力 2)	C (出力)
0 V	0 V	+10 V
0 V	+10 V	+10 V
+10 V	0 V	+10 V
+10 V	+10 V	0 V

(a)　電圧真理値表

A (入力 1)	B (入力 2)	C (出力)	
0	0	1	0 V ≡ 0
0	1	1	+10 V ≡ 1
1	0	1	
1	1	0	

(b)　正の論理真理値表（NAND ゲート）

A (入力 1)	B (入力 2)	C (出力)	
1	1	0	0 V ≡ 1
1	0	0	+10 V ≡ 0
0	1	0	
0	0	1	

(c)　負の論理真理値表（NOR ゲート）

図 6-28　NPN 論理インバータ ゲートの電圧真理値表と論理真理値表

これに対して，実用されている PNP 論理インバータ ゲート（図 6-29 と図 6-30）を調べると，反対の関数になっていることがわかる。正の論理真理値表から，$A'B'=1$ のときに限り $C=1$，すなわち $C=A'B'$ である。ド・モルガンの定理によると，これはまた $C=(A+B)'$ と表わされ，NOR 関数であ

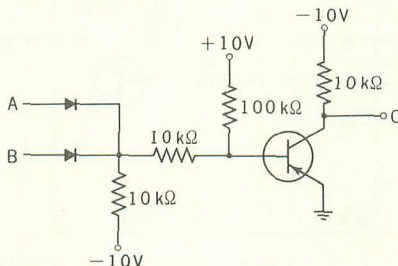



図 6-29　具体的な PNP 論理インバータ ゲート

A (入力 1)	B (入力 2)	C (出力)
0 V	0 V	−10 V
0 V	−10 V	−10 V
−10 V	0 V	−10 V
−10 V	−10 V	0 V

(a)　電圧真理値表

A (入力 1)	B (入力 2)	B (出力)	
1	1	0	0 V ≡ 0
1	0	0	−10 V ≡ 1
0	1	0	
0	0	1	

(b)　正の論理真理値表（NOR ゲート）

A （入力 1）	B （入力 2）	C （出力）	
0	0	1	0 V ≡ 0
0	1	1	−10 V ≡ 1
1	0	1	
1	1	0	

(c) 負の論理真理値表（NAND ゲート）

図 6-30 PNP 論理インバータ ゲートの電圧真理値表と論理真理値表

る。負の論理では，A′B′+A′B+AB′ が 1 のとき，出力は 1 である。再び，A′+B′ と変形され，ド・モルガンの定理によって，C=(AB)′ すなわち NAND 関数である。

NAND ゲートの機能について理解できたので，設計上のいくつかの問題についていくらか詳しく考察してみよう。これは，あとで，インバータの動作原理を基礎としたより複雑な回路を考えるときにも，役に立つだろう。回路の動作の分析は，2 つの場合に分けられる。1 つは，トランジスタを ON にする場合，もう 1 つはそれを OFF にしておく場合である。ここでは，最悪の状態のもとで，ON のときトランジスタを十分に飽和させ，また有害な影響を与えないような I_{co} で十分に OFF にすることを保障するため，トランジスタの電流と電圧がどうなくてはならないかについて考えてみる。

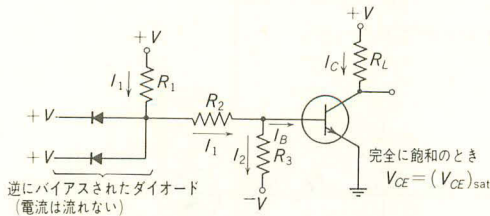



図 6-31 ON 状態での NPN NAND ゲート

ON の状態にある NPN 回路だけを考えると，ゲートへのすべての入力は $+V$ である（PNP 回路は，電圧の極性と電流の方向が逆になっている以外まったく同じである）。図 6-31 に，この論理における電流と電圧を詳しく示してある。まず第 1 に，トランジスタを飽和させるのに必要なだけ，I_C を流すために，I_B は十分な大きさを持っていなければならない。I_B と I_C の一般的な表現は，

$$I_B=I_1-I_2,\quad I_1=\frac{V-(V_{BE})_{\rm on}}{R_1+R_2},\quad I_2=\frac{V+(V_{BE})_{\rm on}}{R_3}$$

$$I_B=\frac{V-(V_{BE})_{on}}{R_1+R_2}-\frac{V+(V_{BE})_{on}}{R_3}$$

$$I_C=\frac{V-(V_{CE})_{sat}}{R_L}$$

飽和するためには，I_C と I_B の比（回路の β*）が，実際のトランジスタの**電流利得**（current gain，h_{FE}）より小でなくてはならない：

$$h_{FE}>\frac{I_C}{I_B}\quad（計算した I_C と I_B を使う）。$$

このゲートの代表的な値を与えて，回路が希望どおり飽和するかどうか，検討してみよう。

$R_1=10\text{k}\Omega$,　　$V=+10\text{V}$

$R_2=10\text{k}\Omega$,　　$(V_{CE})_{sat}=+0.2\text{V}$

$R_3=100\text{k}\Omega$,　　$(V_{BE})_{on}=+0.5\text{V}$

$R_L=10\text{k}\Omega$,　　$h_{FE}=20$

$$I_B=\frac{10-0.5}{(10+10)10^3}-\frac{10+0.5}{(100)10^3}=\frac{9.5}{20}-\frac{10.5}{100}\ (\text{mA})$$

$$=0.475-0.105=0.370(\text{mA})=370\ (\mu\text{A})$$

$$I_C=\frac{10-0.2}{(10)10^3}=\frac{9.8}{10}=0.98\ (\text{mA})\ =980\ (\mu\text{A})$$

検討

$$h_{FE}>\frac{I_C}{I_B}\quad（飽和条件）$$

すなわち，20>980/370？

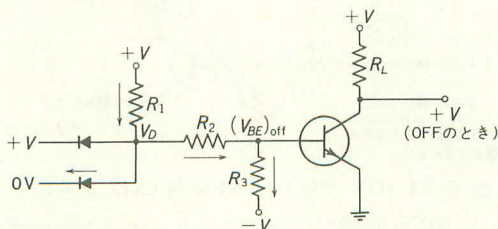



図 6-32　OFF 状態での NPN NAND ゲート

*〔訳注〕 **電流増幅率** β というのは，V_{CE} を一定にして，I_B を横軸にとり，I_C を縦軸にとったグラフを書いたときの傾きのことであり，直線に近いところを，そのトランジスタの電流利得という（右図）。ところが，I_B を大きくしたところでは，I_C/I_B すなわち，電流増幅率は，トランジスタの電流利得より小さくなる。この点を飽和点といい，ここでは I_B をいくら大きくしても I_C は一定となる。

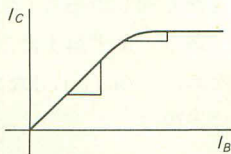
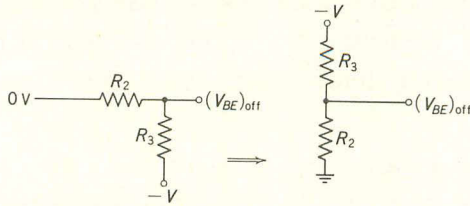



図 6-33　OFF 状態での電圧分割

ゆえに，入力が ON になったとき，回路は正しく飽和する。もし，h_{FE} の小さいトランジスタを使ったり，負荷に対する要求が増加したりしたとき（より多くの I_C が流れるとき），ON の状態で回路が正しく働くかどうか，検討し直す必要がある。最悪の状態は，トランジスタの h_{FE} が最小のときに起こる。h_{FE} は温度によって変化し，温度が下がると小さくなるので，その回路について考えられる最低の温度でも，ON の状態を満足しなくてはならない。

OFF の状態では，振幅が一定値以下の雑音パルスに対してもトランジスタを OFF にしておくように，回路を設計しておかなければならない。この場合の電流と，電圧の記号を 図6-32 に示してある。トランジスタを OFF にしておくには，2つのダイオードに，順方向のバイアスを加えておけばよい。順方向にバイアスのかかったダイオードの両端の電位差は小さい値，V_D である。そして，トランジスタのベースの電圧は R_3 と R_2 の電圧分割によって得られる。$V_D \approx 0$ とみなすと，図 6-33 のように単純化して考えることができる。$(V_{BE})_{\text{off}}$ を求めると，

$$(V_{BE})_{\text{off}} = -\frac{R_2}{R_2+R_3}V$$

となり，前と同じ値を使うと，

$$(V_{BE})_{\text{off}} = -\frac{(10)(10)^3}{(10+100)(10)^3}\cdot 10 = -\frac{100}{110} = -0.91\ \text{(V)}$$

これは，ベース-エミッタ間に，逆方向に約 1V のバイアスがかかっていることを意味し，またトランジスタが OFF になっていることを保証する。トランジスタを ON にするのに $V_{BE}=+0.5\text{V}$ 必要であることから，回路に影響を与えない全雑音電圧は $0.91+0.5=1.41\text{V}$ である。この値以下の電圧の雑音をひろっても，OFF の状態に影響を与えない。トランジスタが OFF でも，少数のキャリアによる電流 I_{co} が R_L を流れるが，それは R_L での電圧降下をもたらすほど大きくてはならない（理想的には $(V_{CE})_{\text{off}}$ は $+V$ ボルト）。I_{co} は

温度上昇につれて増加するので，最悪の場合は，高い動作温度のときに起こる。代表的な $I_{co}=1\mu A$, $R=10k\Omega$ に対して，10V からの電圧降下は $1\times10^{-6}\times10\times10^3=0.01(V)$ すなわち 10mV である。これは 10V と比較して無視できるほど十分小さい。

問題 6-6 次の抵抗と電圧値を用いたとき，図 6-31 のトランジスタを完全に飽和させるのに，$h_{FE}=20$ が十分かどうか調べなさい。ただし，$R_1=2k\Omega$, $R_2=1k\Omega$, $R_3=10k\Omega$, $R_L=1k\Omega$, $V=12V$。

問題 6-7 図 6-32 および 6-33 の $(V_{BE})_{off}$ を 問題 6-6 の値を使って計算しなさい。

直結トランジスタ論理ゲート（DCTL*） トランジスタ論理ゲートは，トランジスタをダイオードのかわりに論理入力素子として使うものである。これはダイオード-トランジスタ ゲート より高速で動作する。トランジスタ論理ゲート（DCTL，直結トランジスタ論理と呼ばれる）の長所は，動作電圧が低いため，ゲートあたりの消費電力が小さく，同時に，集積論理回路技術を使って安いコストで製造できることである。数個のトランジスタを単一操作で以前のトランジスタ 1 個の大きさに作り込むことができるので，多くの入力のある完全なゲ

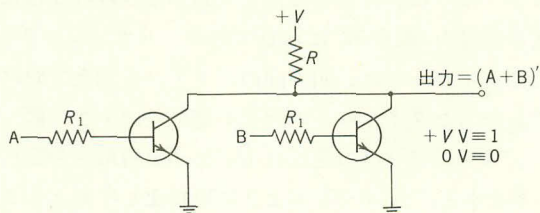



図 6-34 トランジスタ NOR ゲート（正の論理）

A	B	出　力
0V	0V	$+V$V
0V	$+V$V	0V
$+V$V	0V	0V
$+V$V	$+V$V	0V

(a)

A	B	出　力	
0	0	1	$+V$V≡1
0	1	0	0V≡0
1	0	0	
1	1	0	

(b)

図 6-35 トランジスタ NOR ゲート（正の論理）
(a) 電圧真理値表と (b) 論理真理値表

*〔訳注〕 Direct-Coupled Transistor Logic.

ートを1つのパッケージに作ることができる。

トランジスタゲートの1つの型を 図6-34 に示す。それは並列になった2つのスイッチとの類推で説明できる。もし，どちらかのスイッチが閉じられれば（トランジスタ ON），出力は 0V($+(V_{CE})_{\text{sat}}$) である。このように，もしAまたはBが 1 ($+V$) ならば，導通しているトランジスタの出力は低くなる。両方の入力が 0 (0V) のとき，すなわちトランジスタが2つとも OFF のときだけ，出力は $+V$ になる。電圧および論理真理値表（図6-35）は，正の論理では，NOR ゲートであることを示している。負の論理においては，NAND ゲートであることは明らかである。

もう1つの トランジスタ ゲート 回路を 図6-36 に示す。これは正の論理では NAND ゲート（負の論理では NOR ゲート）である。この回路は，直列になったスイッチとの類推で説明できる。どちらかのスイッチが開いていれば，回路は通じない。入力AとBの両方がある（両方のスイッチが閉じている）ときだけ，出力が 0V となる。図 6-37 の電圧および論理真理値表はこの回路の論理演算を表わしている。もしどちらかの入力が 0V なら，そのトランジスタのベースにはトランジスタを ON にする電流が流れず，カットオフのままに

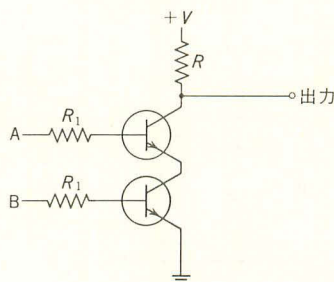



図 6-36　トランジスタ NAND ゲート（正の論理）

A	B	出力
0V	0V	$+V$V
0V	$+V$V	$+V$V
$+V$V	0V	$+V$V
$+V$V	$+V$V	0V

(a)

A	B	出力
0	0	1
0	1	1
1	0	1
1	1	0

$+V$V≡1
0V≡0

(b)

図 6-37　トランジスタ NAND ゲート（正の論理）の
(a) 電圧真理値表と (b) 論理真理値表

なる。両方の入力が $+V$ なら，出力は2つの飽和したコレクタ・エミッタ間の電圧降下の2倍 $(2\times(V_{CE})_{sat})$ に等しい。シリコンエピタキシャルトランジスタでは，それは簡単に1個につき， 0.1V 以下にすることができる。そうすれば， 入力が5つあるゲートで， 0 状態の電圧は +0.5V ということになる。これと区別するのに，1に相当する電圧（水準）として +3V を使えばよい。

トランジスタゲート回路とダイオードトランジスタ論理回路との興味ある相違点は， 前者では NAND 回路も NOR 回路も，同じ型のトランジスタ (PNP または NPN) を使い，同じ電圧を入力として使えるから，同じシステムの中で互換性を持って使えることである。ダイオードを使っている NAND および NOR ゲートでは，反対の型のトランジスタ（正または負の論理のどちらかに対して）を使い，異なった論理電圧水準を用いる必要がある。

半加算器を作るのに，トランジスタゲートがどのように使われているかの例を 図6-38 に示す。和と桁上りの表現は次のようであった。

$$和=AB'+A'B=(AB+A'B')'$$

$$桁上り=AB$$

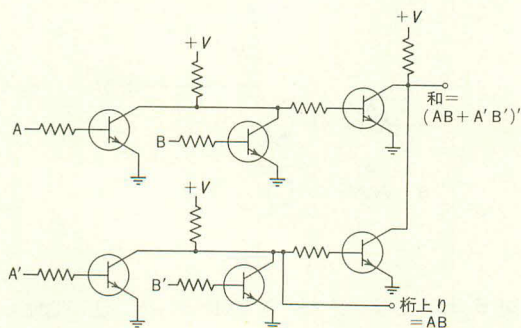



図 6-38　トランジスタゲートを用いた半加算器

問題 6-8　ブール関数　AB′+A′BC を作る DCTL 回路を書きなさい。
問題 6-9　ブール関数　(A+B)AB′C′ を作る DCTL 回路を書きなさい。

要　約

この章では，AND, OR, INVERTER, NAND および NOR ゲートを説明した。AND と OR ゲートは論理関数を作るために使われるが，負荷の問題

で，その使用は制限される。設計者は，論理関数を作るために NAND または NOR ゲートをより多く使用する。なぜなら，信号の振幅と速度が負荷に及ぼす影響がはっきり定義できるからである。マイクロ エレクトロニクス の出現は回路の標準化を促した。

問　題

1. +10V と 0V を電圧水準として，正の論理の OR ゲートの回路図を書き，その電圧および論理真理値表を書きなさい。

2. 0V と −10V を電圧水準として，正の論理の OR ゲートの電圧および論理真理値表を書きなさい。

3. 問題 6-3 を，$R_1=1\,\mathrm{k\Omega}$ および $R_2=5\,\mathrm{k\Omega}$ について，計算し直しなさい。

4. 負の論理の ダイオード トランジスタ NOR ゲートの回路を書きなさい。

5. 負の論理の NAND ゲートの電圧および論理真理値表を書きなさい。

6. $R_1=5\,\mathrm{k\Omega}$, $R_2=2\,\mathrm{k\Omega}$, $R_3=12\,\mathrm{k\Omega}$, および $R_L=500\Omega$ として，問題 6-6 を計算しなさい。ただし他の値は変更しない。

7. 問題6 の値を使って，問題 6-7 を計算しなさい。

8. +10V と 0V を論理水準として，NOR ゲートの電圧および論理真理値表を書きなさい。正，負どちらの論理を使っているか記しなさい。

9. 図 6-39 の NAND ゲートで，電圧真理値表を書きなさい。

10. 論理式 $W=X'YZ+XZ'+YZ$ を作る DCTL 回路を書きなさい。

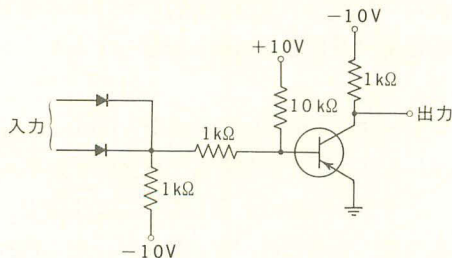



図 6-39　問題 6-9 の NAND ゲート回路

0111 ══ 7

電子計算機の回路

6章で計算機の論理回路について述べてきたが，これから論じることはマルチバイブレータ回路についてである。この回路の中で，最も一般的なものは**2安定マルチバイブレータ** (bistable multivibrator) または **フリップ フロップ** (flip-flop) と呼ばれるもので，記憶素子，カウンタ（計数器）あるいはシフト レジスタ に使うことができる。論理回路はブール代数式で表わされた論理演算を行なうためのものであるが，2 安定回路は，データをしまったり，ある番地から別の番地へデータを移したりすることを並列に（すべてのビットを一度に）行なったり，直列に（1 ビットずつ）行なったりすることができる。2 安定回路は，また，算術演算（計数した結果を累算しておく）を行なうために計数器の中でも使われる。

1 安定マルチバイブレータ (monostable multivibrator) またはワンショット (one-shot) と呼ばれるものは，タイミングを合わせたり，パルスを整形したりするのに使われ，トリガ (trigger) 信号が入ってくると，一定の長さのパルスを作り出す働きをする。そのため，ある動作が一定の遅れのあとに行なわれなければならない場合とか，パルスの幅を長くしたり短くしたりしなければならない場合に使われる。

無安定マルチバイブレータ (astable multivibrator) またはクロック (clock) と呼ばれるものは，一定の周波数の矩形波を作り出し，計算機が演算を行なう時間を制御するのに使われる。この章では，この3つのマルチバイブレータの類似点，相違点について詳しく述べる。基本的な応用については，8章で論じる。

終りに，**シュミット トリガ** (Schmidt trigger) **回路**と呼ばれるものについて

詳しい説明とその応用について述べる。シュミットトリガ回路は，ゆっくりと変化する波形を，ディジタル回路に適するように，急激に上昇したりあるいは下降する傾斜を持つ波に変換する整形回路である。

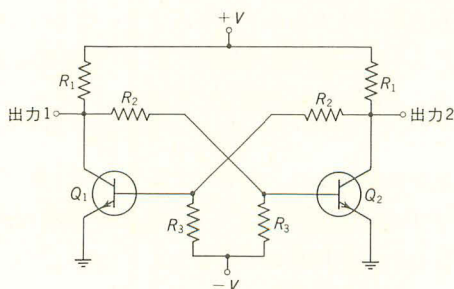



図 7-1　マルチバイブレータ回路

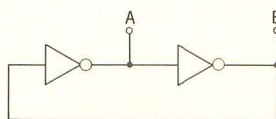



図 7-2　2つのインバータによる2安定回路

7-1　2安定マルチバイブレータ回路

基本的なマルチバイブレータの回路を 図7-1 に示す。この回路は，2つの導通状態のいずれか一方で安定している。もし，Q_1 が導通状態にあり，Q_2 がカットオフの状態（非導通状態）にあるならば，回路は**トリガ信号***が入ってくるまでそのままの状態にある（電源は切らないとする）。もし，Q_2 が導通状態で，Q_1 がカットオフの状態になっていれば，この状態はやはりそのまま続く。この2つの可能な安定状態，すなわち，2 安定状態は，この場合，抵抗 R_2 でできている交さした結線によって生じる。もし，交さした素子のうち，一方がコンデンサであったら，回路は1つの安定な状態に維持することだけしかできないし，もし両方の素子がコンデンサだったら，回路には安定な状態はなく，交互発振して無安定な動作になる。

*〔訳注〕エネルギの小さな制御信号で，大きなエネルギを出すこと，あるいは，このような動作をする回路のことで，エネルギの小さな制御信号のことをトリガ信号という。

2安定装置の働きを理解するために，図7-2の回路を考える。2つのインバータがループして結ばれている。1番目のインバータへの入力Bが1ならば，この出力Aは0となる。この0は2番目のインバータへの入力であり，出力として1を出す。これは，明らかに意図したとおりのものである。そこで，最初のインバータへの入力Bが0であるとしてみよう。このインバータの出力Aは1となり，次のインバータからの出力は0になって，最初に入れた0と一致し，この状態は安定する。上のいずれの場合にも安定してしまう。上の各状態を安定状態という。

AとBとは常に互いに反対になっていて，A と A′，真（TRUE）と偽（FALSE），または1と0などのようになっていることは明らかである。

2安定回路は，2つの安定状態のうちどちらかになっている。この安定状態を反転して，もう一方の安定状態にするにはどうしたらよいだろうか。これを理解するために，インバータのかわりに，図 7-3 のような2つの NOR ゲートを使って考えてみよう。

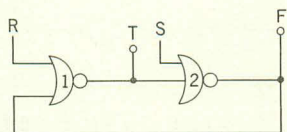



図 7-3　SET (S) および RESET (R) の入力がある2安定回路

入力RとSが偽，すなわち0ならば，いままで述べてきたインバータだけの場合と同じである。T での出力が，1 の状態になっていると仮定する（Fは当然0となる)。T の出力が1になっている場合のことを，SET 状態であると定義する。また，F の出力が1になっていることを，回路は RESET 状態であると定義する。ここに図示してあるR と S はそれぞれ RESET または SET 状態にする入力である。S が1になると，T は1となる。すなわち，この回路を SET の状態にする。ここでRとSが同時に1となりえない（また，そうなっていない）ものとすると，R は0となっている。NOR ゲート1の出力Tが1であるから出力 F は0となる（NOR ゲート2の入力に入いるT信号によって)。SET 信号が変化（S が0に戻る）しても，NOR ゲート1への入力としての出力 F は，出力 T を1にする。すなわち，SET 状態に安定している。2 安定回路を RESET するために，入力Rを1にすると，出力Fは 1 になり，出力Tは0となる。R が0に戻っても，回路はそのまま RESET 状態（出力 F が 1，T が 0）を保ち続ける。

まとめて考えると，2 安定回路の 2 つの出力を T(true) および F(false) と定め，2 つの安定状態を，SET および RESET と定めることは自然である。ここで，SET というのは，T=1，F=0 の状態のことで，RESET というのは，T=0，F=1 の状態のことであると定義している。

さて，この一般的な事柄を念頭において，2 安定回路の詳細に話を戻そう。2 つの NOR を使った回路が正しく動作していても，それは直流電圧レベルで働いているにすぎない。もっと融通性のある動作をさせるためには，交流またはパルスの入力でも動作すべきであり，それに対するもっと詳しい回路について考えよう。

図 7-1 の 2 安定回路は対称であって，Q_1 を導通状態に保つための R_1, R_2, R_3 の回路は Q_2 を導通状態に保つためのものと同じである。同様に，Q_1 をカットオフに保つための回路部分は，もう一方の安定状態で Q_2 をカットオフの状態にしておく回路部分と同じである。Q_1 が導通で，Q_2 がカットオフのときの動作を分析すれば，もう一方の Q_2 が導通で Q_1 がカットオフのときの動作状態も分析されたことになる。

Q_1 が導通状態にあれば，コレクタの電圧は飽和電圧になっていて，約 0.5 V

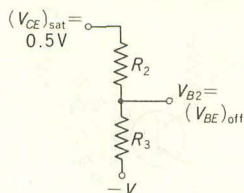



図 7-4　トランジスタのカットオフ状態での部分回路

である。この状態では，Q_2 のベース電圧は R_2 と R_3 とによる電圧分割で決まる。この状態の回路の一部を 図 7-4 に示す。

$(V_{CE})_{sat}=0$ V と仮定して，V_{B2} の近似値を求めると，次のようになる。

$$V_{B2}=-\frac{R_2}{R_2+R_3}V$$

一般に，$R_2 \ll R_3$, $V \gg V_{sat}$ であって，Q_2 のベース電圧が負（Q_1 が導通）となり，Q_2 はカットオフすなわち非導通の状態にある。つづいて，Q_2 がカットオフの状態のとき，Q_1 のベース電圧を決めることができる。図 7-5 を見ると，Q_1 を導通にするためには，ベース電圧は $(V_{BE})_{on}$，または近似的に 0.7 V でなければならない。トランジスタを導通にする重要な要素は，これを飽和状態にすることである。ベース駆動電流は，$I_B=I_1-I_2$ である。ここで，

$$I_1=\frac{V-0.7}{R_1+R_2},\quad I_2=\frac{V+0.7}{R_3}$$

である。

これらの式から，R_1 と R_2 は，十分なベース駆動電流が流れるのに必要なだけ小さくなければならないし，一方，ベース以外の方へ逃げてしまわないため，R_3 は大きな値でなければならないことがわかる。飽和させるためには，ベース電流は I_C/β $(\beta I_B=I_C)$ より大きくなければならない。このような条件さえ満たされるならば，この回路（図 7-1）は2安定回路として満足に働くことになる。一方のトランジスタが導通の状態になっている限り，それによって，電流は R_2, R_3 を通って $-V$ のところへぬけ，もう一方のトランジスタを非導通にする。また一方のトランジスタが非導通になっていれば，$+V$ から R_1 と R_2 を通して，導通のトランジスタに十分なベース駆動電流を与える。R_1, R_2, R_3 を選ぶことについて，もう1つ別の基準が考えられる。非導通のトランジスタのコレクタに高い電圧を与えるためには，$+V$ と $(V_{BE})_{\rm on}$ との間の電圧分割によって適正な電圧が得られるように，R_2 は R_1 より大きくなければならない。図 7-6 の部分回路から，$(V_{BE})_{\rm on}\approx 0$ と仮定すると，V_{C2} の近

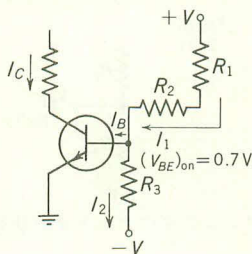



図 7-5 トランジスタが導通になっているときの回路の一部

V
R_1
R_2
V_C2=V_off
Q_2
OFF
(V_BE)_on

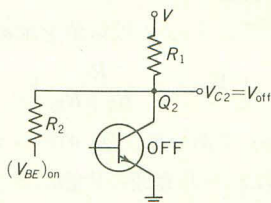



図 7-6 $V_{\rm off}$ の説明用の部分回路

似値は，次のように表わされる。

$$V_{C2}=\frac{R_2}{R_1+R_2}V$$

補数をとる，すなわち，**フリップ フロップ**の状態を逆にするためには，アース電圧（0 V）を非導通のトランジスタのコレクタまたは導通トランジスタのベースに与えればよいだろう。アース電圧がとり去られても，回路はこの逆転した状態のまま保たれる。このように，2 安定回路は，主に記憶素子として有用である。すなわち，最後に入った状態を“記憶”しているのである。もっと一般的に，この回路を計数回路やシフト回路として使用するためには，それらの入力回路にトリガ回路を用いる。

導通トランジスタをカットオフ状態にするためには，負の信号がベースに入ればよい。不必要に信号を入れて，回路を異常な状態にしないように，うまくベースにパルスを入れるには，静電容量結合を用いる。基本的な入力回路は，一方の極性のパルスをカットするのにダイオードを用いた微分回路*である(図 7-7)。これを入力回路として使うと，導通トランジスタを OFF にするための

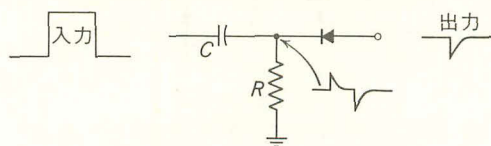



図 7-7　マルチバイブレータへの入力回路

負の方向の電圧変化だけが入っていくことになる。直流信号の径路でコレクタやベースを接地し，パルスが入っている間，回路を一定の状態に保っているのに対し，この入力回路では交流径路を通して（信号変化に対応して）カットオフの信号を与えることになり，入力回路自体の状態は一定しているわけではない。負のパルス幅は導通トランジスタを OFF にするのに必要なだけの長さがあればよい。この形で，交流の SET および RESET 信号を使うと，フリップ フロップは適当な制御端子（SET または RESET）へパルスを与えることによって制御されることになる。しかし，両方へ同時にパルスが入ると，両方のトランジスタにカットオフのパルスが与えられ，回路は正しく動作しない。

1つのパルスがフリップ フロップを逆の状態にするような計数動作では，導通トランジスタを OFF にするだけであるから，1つの**方向づけ回路**（steering network）を用いなければならない。非導通トランジスタは，フリップフ

---

*〔訳注〕 C-R 回路による微分回路で R の両端に生じる電圧は $E(1-e^{-t/RC})$ の波形を持つ。時定数 RC（秒）が十分小さければ波形は 図 7-7 に示すように鋭いスパイクを持つ。この形が入力パルス電圧波形を微分した形に似ているのでこの名前がある。

ロップの交さ回路により ON になる。フリップフロップの状態を“検知”し，負のパルスを導通トランジスタだけに送り込むために，方向づけ回路の各抵抗はトランジスタのコレクタに帰路を持つように接続される。これを図 7-8 に示してあるが，ここではフリップフロップの部分は枠の内に入っており，方向づけ回路は枠の外に出ている。

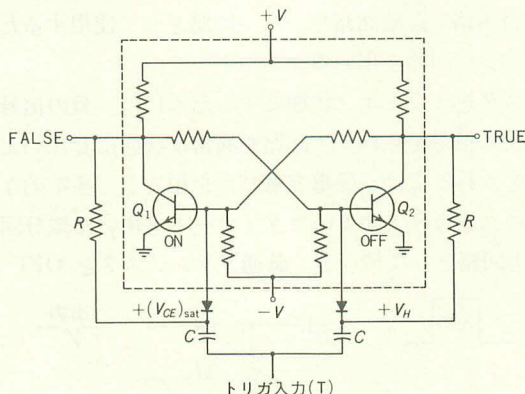



図 7-8　方向づけ回路のあるフリップフロップ

各トランジスタへの入力として，抵抗，静電容量，ダイオードの結線を考えると，導通トランジスタにつながっている静電容量は，$(V_{CE})_{\mathrm{sat}}$ に充電されており，もう一方の静電容量は V_H に充電されている。信号の大きさを回路が正しく働くように選ぶと（$V_{\mathrm{trigger}}=V_H$），導通トランジスタだけが負のパルスを受ける。トリガパルスが正になるときには，その正の微分信号はベースにつながれているダイオードによって妨げられる。トリガ信号の下降の段階では，非導通トランジスタに関係している静電容量にかかる電圧は V_H V だけ変化する。しかし，この電圧は，はじめに V_H V であったのだから，0 V に下がることになる。非導通トランジスタのベースは数ボルトだけ負になっているから，ダイオードは逆バイアスのままであり，パルスはベースに現われない。導通トランジスタの方では，ベースの電圧ははじめ $(V_{BE})_{\mathrm{on}}$ であったのだから，負のステップ電圧 V_H によってダイオードは導通になり，その結果，負のパルスがベースに与えられ，導通トランジスタは OFF になる。導通トランジスタだけが電流を遮断する負の電圧を受ける。これが OFF になると，非導通トランジスタは，交さ回路によって ON にかわり，この回路の逆転が完了することになる。次のトリガパルスで，方向づけ回路が反対にチャージ（再び非導通の方に V_H）されるにつれて，回路は再び反対の状態にかわり，この

2安定回路は前の状態に戻る。このように，トリガ パルス はパルスが下降するときに，フリップ フロップ を逆の状態にする。これは計数するのに使うことができる。

方向づけ回路は，ディジタル システム では他の重要な動作に関連して使われることがある。データ（固定長語のビット）を計算機の中でシフト* することとは，しばしば，非常に有用なことである。2 安定回路は1つのシフト段**として使われ，多くのシフト段をつないで シフト レジスタ を形成する。方向づけをする回路は，図 7-8 に示すように結線すると，導通トランジスタのベースに入力の トリガ パルス が直接に入る。もし，抵抗 R が他の段のコレクタにつないであると，方向づけ回路はこの他段の回路のコレクタ電圧で方向づけられることになる。もし，この他段の2安定回路で TRUE 側から出力1を出し，FALSE 側から出力0を出しておれば，シフト パルス がトリガ入力に入ってきたとき，これらの信号は方向づけをする抵抗 R につなぐことにより，この段の状態を他の段の状態に進めることができる。もし他の段の出力が，FALSE=1, TRUE=0 であれば，シフト パルス が入ってくれば，他段の状態がこの段の状態へ強制的に移る。言いかえると，シフト パルス によって1つの段から2番目の段へ，1つの段に記憶してあった情報をシフトしたということである。この例は8章に述べてある。

問題 7-1　PNP トランジスタを使って，フリップ フロップ の回路図を書きなさい。

問題 7-2　図 7-1 のような フリップ フロップ 回路について，OFF トランジスタのベース電圧とコレクタ電圧を計算しなさい。ただし，

$R_1=1\,\text{k}\Omega$　　$R_2=2\,\text{k}\Omega$　　$R_3=10\,\text{k}\Omega$　　$V=+10\text{V}$

である。

問題 7-3　方向づけ回路を含む PNP フリップ フロップ の回路図を書きなさい。

7-2　1 安定マルチバイブレータ回路

マルチバイブレータのうち，1安定回路というのは，1つの安定状態を持つものである。すなわち，一方のトランジスタが ON で，もう一方が OFF である状態は2通り考えられるが，そのうち1つだけが安定なものをいう。たとえば，Q_1=OFF, Q_2=ON というのが安定状態だと定めよう。この回路に信号が入らない限り，いつまでもこのままの状態が続く。入力パルスが入ると，状態

*〔訳注〕レジスタにある情報（2進数表示）の桁位置をずらすこと。

**〔訳注〕シフト段 (shift stage) は1ビットを記憶するフリップ フロップ である。

は Q_1=ON, Q_2=OFF にかわる。しかし，この状態は安定ではなく，回路は一定の期間だけこの状態を保っているにすぎない。これは数分の1マイクロ秒から，数秒の間である。この期間が終ると,回路は再びもとの安定状態に戻り，またトリガパルスが入ってくるまでいつまでもそのままである。図7-9の1安定回路は，基本的なマルチバイブレータ回路において交さ抵抗のうちの1つを静電容量で置きかえたものである。このほか，抵抗 R を $+V$ のところに接続することによって，2安定回路を1安定回路にかえている。

図に示してあるトリガ入力回路は,与える入力パルスを微分する回路である。ダイオードは，正方向の信号が Q_2 のベースへ入るのを妨げる。ベースへは負の信号だけが伝わる。Q_2 は導通状態のトランジスタであり，安定状態にあるから，この入力パルスによって Q_2 は OFF になる。

安定動作状態では,トランジスタ Q_1 は OFF で Q_2 が ON である。トリガパルスが入って，Q_2 のベース電圧を負にし，Q_2 を OFF にすると，Q_2 のコレクタに接続している交さ回路により,Q_1 は ON になる。かくて Q_2 を OFF にするパルスは，Q_1 を ON にするのに十分な長さだけあればよい。Q_1 が ON になると，静電容量 C は放電し，その結果は負の階段状電圧により Q_2 のベース電圧は同じ量だけ負となる。すなわち，静電容量両端の電圧は瞬間的に変化することができないで，静電容量の Q_1 側に加えられた負の階段状電圧に追

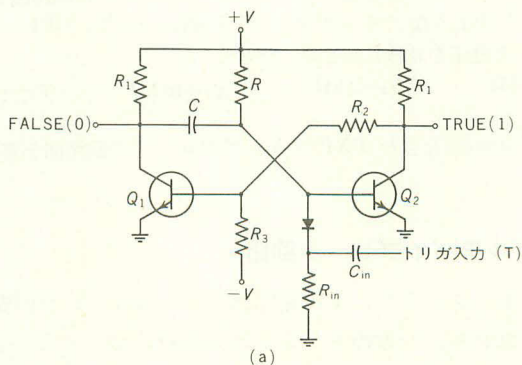



図 7-9　1安定マルチイバブレータ：(a) 回路図　(b) ブロック図

随して静電容量の Q_2 側の階段状電圧が同じ振幅を持って負方向に変化する。すなわち，Q_2 のベース電圧は負となるわけである。次に，静電容量 C は時定数 RC で $+V$ に充電されることになる。この Q_2 側の電圧が $(V_{BE})_{on}$ に達すると，Q_2 は ON に戻る。Q_2 が飽和し，R_2 のコレクタ側が接地電圧近くになると，R_3 と R_2 の電圧分割によって Q_1 のベースの電圧は負になり，Q_1 は OFF になる。1安定回路の働きは 図 7-10 に示す波形によって理解できる。不安定状態にある期間は RC によって決まり，近似的に $0.7RC$ 秒である。（この値は，$+V$ と $-V$ の大きさが等しい限り，かなりよい近似値である。）

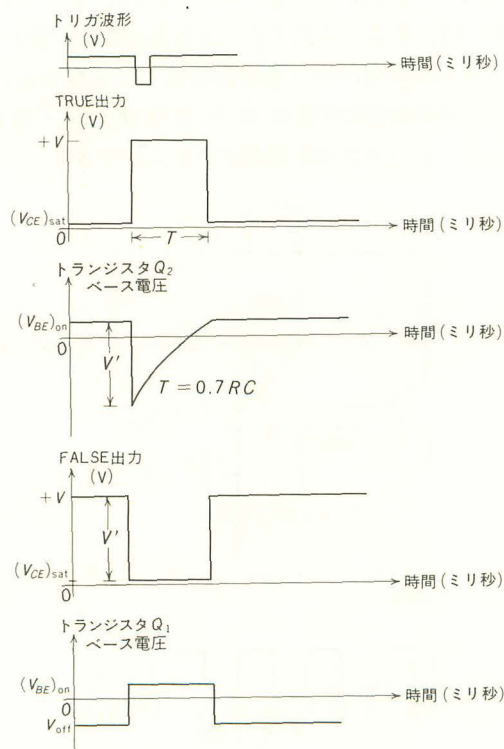



図 7-10　1安定回路の波形

計算機の回路において，1安定マルチバイブレータは，主にパルスの波形整形や信号のタイミングの調整，あるいは一定の時間遅れを作るのに使われる。TRUE 側の出力は，一定の幅を持つ正のパルスである。いろいろなパルス幅

を持つ信号が入ってきても，1安定回路の出力パルスの幅は一定になる。たとえば，論理ゲートにいくつかの信号パルスが入ってきて，それらのパルスが時間的に正確に一致していなかったとすると，このゲートから出てくるパルスは一定ではない。出てきたパルスの長さが一定でなくても，1安定回路を通した結果，一定の幅のパルス，すなわちパルス時間一定のパルスが得られる。もう1つの例は，記憶装置からの出力である。この場合，読出し回路から出てくるパルスは，狭くて貧弱な波形である。このパルスを1安定回路でもっと長く作り直すと，計算機の他の回路で使うのにより有効な信号が得られる。一定周波数のパルス波が入ると，1安定回路からの出力は同じ周波数となるが，出力信号の**デューティ サイクル**（duty cycle,　OFFの時間/ONの時間）は，1安定時間の長さ $T=0.7RC$ によって決まる。信号周波数に影響を与えないでいろいろなデューティ サイクル のパルス波を出せるパルス発生器を作るのに，この時定数が使われる。回路の中に可変の R と C を使えば，望ましい デューティサイクル になるようにパルス幅を調整することができる。

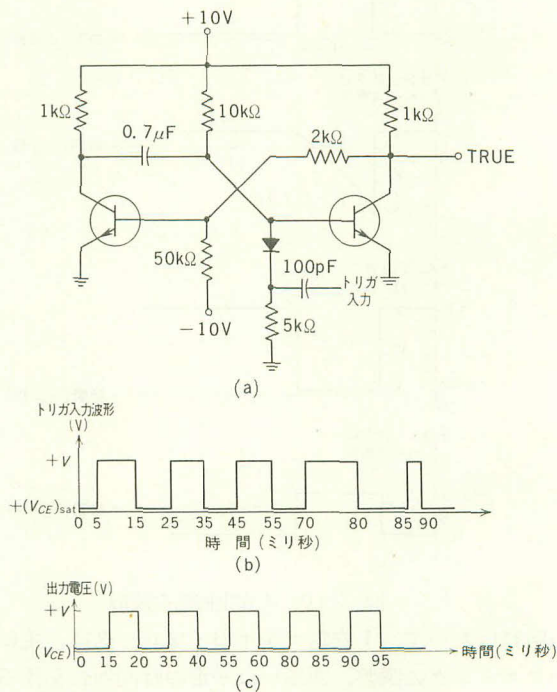



図 7-11　例 7-1 の1安定回路と波形

ここで示した NPN 型の回路では，1安定回路は負の方向への電圧変化でトリガされる。TRUE のコレクタ側からの出力は，トランジスタが飽和しているとき（安定の状態のとき）の約0 V から，1 安定のパルス時間での約 $+V$V に変化する。NPN 型と PNP 型の1安定回路がどのように働くか，次に 2 つの例をあげる。

例 7-1 次の NPN 型の1安定マルチバイブレータ（図 7-11）について，与えられた入力波形に対する出力波形を書きなさい。

解：まず，1安定回路の時定数 $T=0.7RC$ を計算する。R は 10kΩ, C は 0.7μF であるから，

$$T=0.7(10\times10^{3})(0.7\times10^{-6})\ (秒)$$
$$=0.49\times10^{-2}=4.9\times10^{-3}$$
$$\approx5\ (ミリ秒)$$

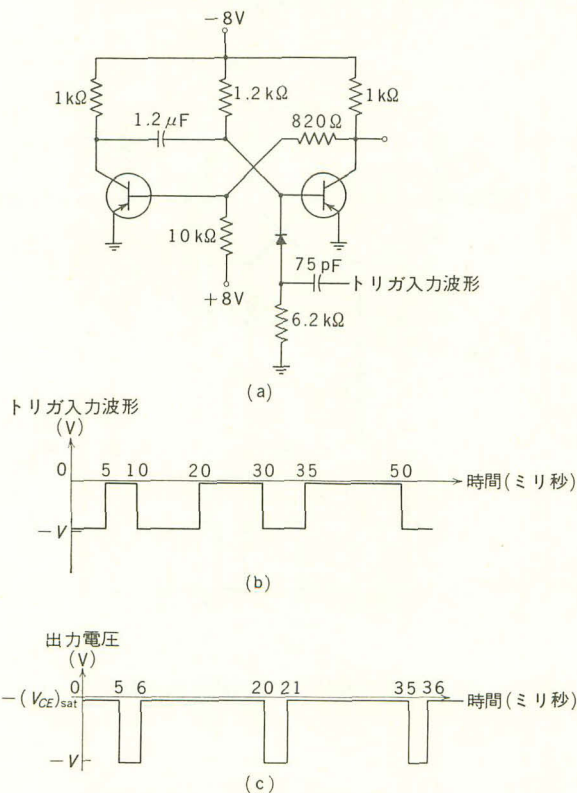



図 7-12　例 7-2 の1安定回路と波形

入力が負方向へ移行するときに毎回トリガされるから，結果の出力波形は，図 7-11(c) のようになる。

例　7-2　与えられた PNP 型の回路（図 7-12）について，示されたトリガ入力信号に対する出力波形を書きなさい。

解：時定数 $T=0.7RC$ は

$$T=0.7\times(1.2\times10^{3})(1.2\times10^{-6})=1\ （ミリ秒）$$

入力パルスが正方向へ移行するときに，トリガされる（PNP 回路）。ゆえに，出力波形は 図 7-12(c) のようになる。

問題 7-4　図 7-13 の回路について，示された入力波形に対する TRUE 出力波形を書きなさい。

問題 7-5　図 7-14 の回路に，1 kHz のパルス波が入ったときの FALSE 側の出力波形を書きなさい。

問題 7-6　図 7-15 の回路の 1 安定時定数を計算しなさい。

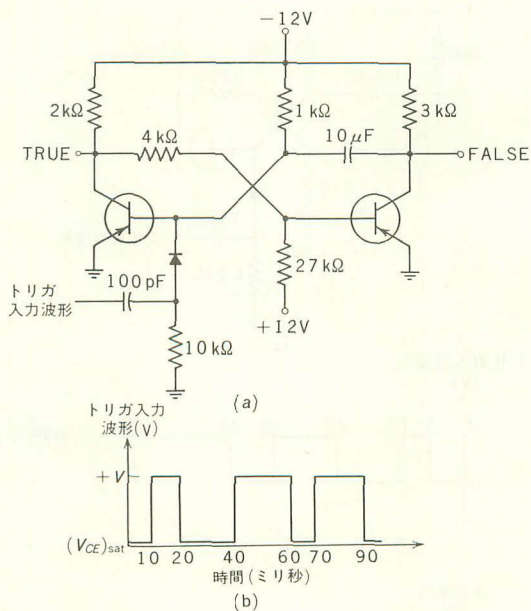



図 7-13　問題 7-4 の回路と入力波形

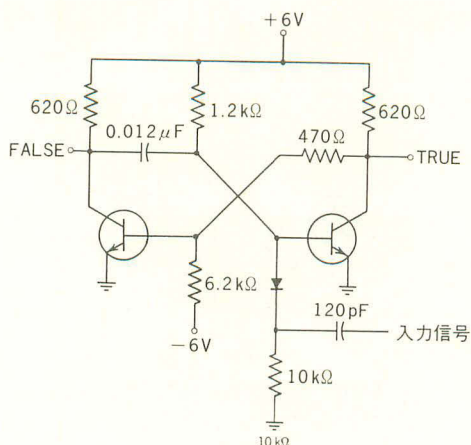



図 7-14 問題 7-5 の回路

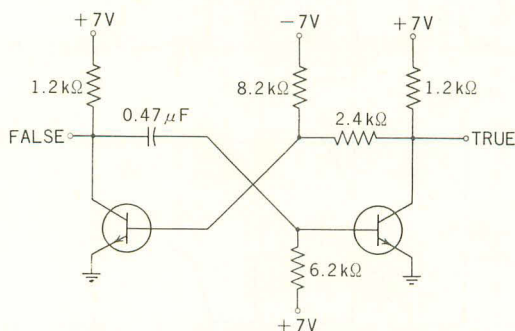



図 7-15 問題 7-6 の1安定回路

7-3 無安定マルチバイブレータ回路

計算機やディジタル回路の動作の中の重要な部分は刻時（タイミング）信号である。システムのすべての動作を制御するために基本になる発振器があって，刻時信号を出している。刻時信号を得るための1つの簡単な方法は安定状態のないように設計されたマルチバイブレータを使うことである。

これは2つの半安定状態（準安定状態）の間で交番するわけで，この交番する回数は，マルチバイブレータ回路の刻時周波数となる。このような回路を「無安定マルチバイブレータ」と言い，図 7-16 にその回路図を示してある。この図によると前の基本的なマルチバイブレータの交さ素子が静電容量 C_1 と

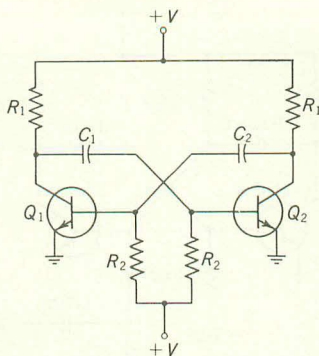



図 7-16　無安定マルチバイブレータ

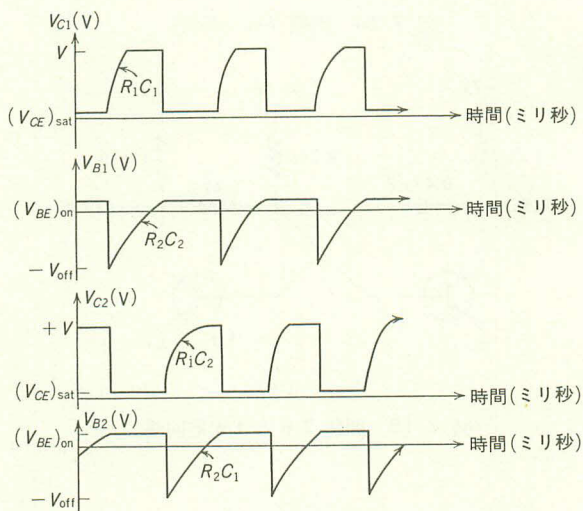



図 7-17　無安定マルチバイブレータの波形

C_2 になっている。交さ素子に静電容量を使うと，一定の時間間隔だけ安定状態（ON または OFF）になる。1 安定回路では，交さ回路に 1 つだけ静電容量が入っていることで，一方の状態だけ安定であるが，無安定回路では 2 つの交さ静電容量があるため安定状態になることができず，出力としては一定の時間だけどちらかの状態になっている。この時間は数分の 1 マイクロ秒から数秒の間である。この出力を，（外から信号が入らない限り）永久に一定の状態を続ける 2 安定回路や，トリガが入ってきたとき，一定の時間だけ他の状態になり，再びもとに戻る 1 安定マルチバイブレータの出力と比較してみよ。

無安定回路の働きを解析するために，はじめ，Q_1 が導通状態（ON）で，Q_2 が非導通状態（OFF）になっていると考えよう。この状態ではコレクタの電圧は $V_{C1}=(V_{CE})_{\mathrm{sat}}$, $V_{C2}=+V$ である。静電容量 C_1 は導通トランジスタ Q_1 を通して接地されているので，抵抗 R_2 を通して C_1 は充電され，充電の時定数は R_2C_1 である。V_{B2}（Q_2 のベース電位）が $(V_{BE})_{\mathrm{on}}$ （≈ 0.7 V）に達すると，Q_2 は ON にかわる。C_2 はトランジスタを通して放電するが，静電容量両端の電圧は直ちに変化することはないので，Q_1 のベース電位は負となる。Q_1 が ON のときのベース電圧，すなわち，$(V_{BE})_{\mathrm{on}}$ からステップ状に負の電圧がかかるから，ON であった Q_1 は OFF にかわる。かくして Q_1 と Q_2 は，一時的にその状態を交換したわけで，今度は静電容量 C_2 は Q_1 のベース電圧が，はじめの負から $(V_{BE})_{\mathrm{on}}$ になるまで，もう一方の R_2 によって充電されることになる。充電する時定数は R_2C_2 である。充電している間は，両方の出力は変化しない。充電が終ると，Q_1 を通って C_1 が放電し，Q_2 が OFF になり，状態は再び交換される。2 つのトランジスタのベースとコレクタの波形を図 7-17 に示す。この図は各状態の時間の長さを制御している充電の過程を図解的によく説明している。$C_1=C_2$ ならば，波形は対称で，出力は矩形波になる。各状態の時間は近似的に $T=0.7R_2C$ である。1 サイクルはこの倍の時間であり，$f=1/(2T)$ であるから繰り返し周波数は次式で計算できる。

$$f=\frac{1}{2(0.7R_2C)}=\frac{1}{1.4R_2C}$$

静電容量の容量 C を増せば周波数 f は小さくなり，C を減らせば f は大きくなるので，周波数を正しく調節することができる。

図 7-17 の波形を見ると，マルチバイブレータの出力は期待通りいつも互いに反対になっている。書き込んであるコレクタの波形の時定数は，R_1 が R_2 より小さくて，コレクタの波形の方が速く $+V$ に上がっていることを示している。もしトランジスタが再び ON になるまでにコレクタが $+V$ に達しないような抵抗 R_1 を採用すると，上の周波数の式はあてはまらず，非常に低い周波数を与えることになる。普通の動作状態ではコレクタは一定の電圧レベルに早く到達し，前に述べたように回路が働いて発振周波数は上の式で決まる。必要ならば，発振器を外部信号に同期させることができる。この場合，外部信号が負の方向のパルスであれば発振器のコレクタへ，または正方向のパルスであればベースに加えればよい。

例　7-3　次の PNP 無安定発振器（図 7-18）の周波数を計算しなさい。

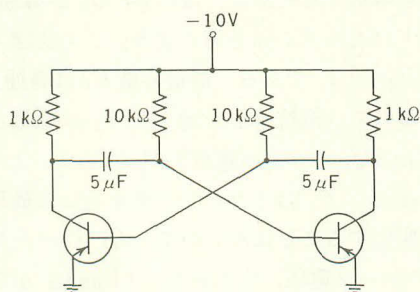



図 7-18　例 7-3 の無安定マルチバイブレータ

解：時定数を決めるものは，$R=10\mathrm{k}\Omega$, $C=5\mu\mathrm{F}$ である。周波数の式　$f=1/1.4RC$ より，

$$f=\frac{1}{1.4(10\times10^{3})(5\times10^{-6})}$$

$$=\frac{1}{7\times10^{-2}}=\frac{100}{7}\fallingdotseq14.28\ (\mathrm{Hz})$$

+8V
6.2kΩ
6.2kΩ
320pF
320pF
9.1kΩ
9.1kΩ
+8V

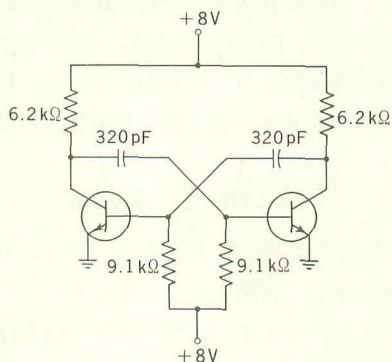



図 7-19　問題 7-7 の無安定回路

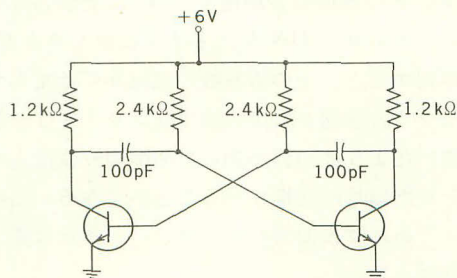



図 7-20　問題 7-8 の無安定回路

問題 7-7 図 7-19 の回路の周波数を計算しなさい。

問題 7-8 図 7-20 の回路の周波数を計算しなさい。

7-4 シュミット トリガ

シュミット トリガ（図 7-21）は再生増幅器としての動作をし，ゆっくりと変化する波形の入力に対してパルス整形を行なうものである。入力電圧（E_{in}）がある値（E_{on} この値は回路素子によって決まる）を越えると，入力のトランジスタは ONになり，出力側のトランジスタが OFF になり始める。この動作は再生的である。すなわち，出力が OFF になっていくと，入力側をさらに ON にする。入力が ON になるということは，出力側をもっと速く OFF にすることになる。これは，出力にパルス波（上昇，下降時間の速い波形）を作ったことになる。次に入力信号が第2の閾値（E_{off}，これも回路素子によって決まる）より下がると，入力側のトランジスタは切れ，出力側を ON にし，これよりさらに入力側が OFF になるようになる。正弦波入力の代表的な場合の入力および出力波形を 図 7-22 に示す。

入力が 60 Hz の正弦波（ミリ秒の単位でゆっくりと上がったり下がったりする波）ならば，出力側の上昇および下降時間はマイクロ秒の何分の 1 かである。出力は入力と同じ周波数を持ち，鋭い波形になっているために，ディジタル回路に適している。特殊な場合として，ON および OFF にする閾値を両方とも 0V にするならば，正弦波の入力に対する出力波形は，これと同じ周波数の矩形波になる。

この詳しい説明を 図 7-21 を参照しながら行なおう。シュミット回路に入力

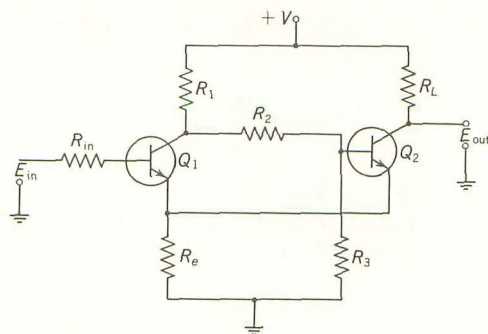



図 7-21 シュミット トリガ回路

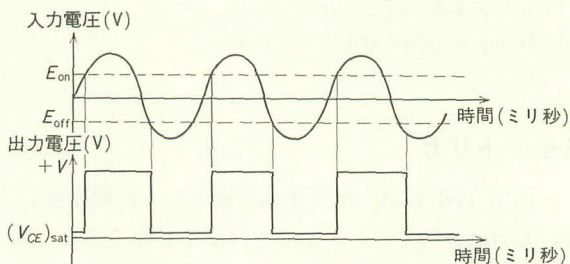



図 7-22　シュミット トリガ 波形

がないと，Q_1 は OFF, Q_2 は ON になっている。Q_2 に関する電圧水準を解析するのに，回路の一部をとり出してみるとわかりやすい。図 7-23 にこの部分回路を示す。一般に，Q_2 のベース電圧は，R_e を負荷抵抗とした**エミッタ ホ**

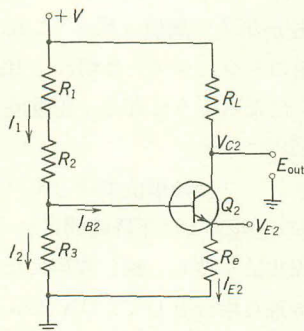
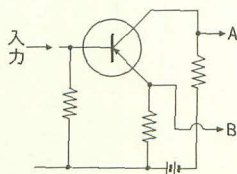



図 7-23　シュミット トリガ の部分的な出力回路

ロワ (emitter follower)* のトランジスタ Q_2 の入力抵抗と R_3 の並列抵抗と，R_1 と R_2 の直列抵抗とで分割された電圧となるであろう。R_e の値をトランジスタの電流増幅率倍したものが R_3 に比べて十分大きいならば，ベース電圧は簡単に次式で表わされる。

$$V_{B2}=\frac{R_3}{R_1+R_2+R_3}V$$

*〔訳注〕 右図のようなエミッタ接地の回路では，普通，出力を A からとるが，B のようなところからとる方式をエミッタホロワ方式という。

$V_{BE2}=V_{B2}-V_{E2}$ であるから，$V_{E2}=V_{B2}-V_{BE2}$ となる。導通トランジスタの V_{BE2} は，近似的に 0.7V であるから，$R_3/(R_1+R_2+R_3)$ という抵抗の比は導通トランジスタのエミッタ電圧を決めることになる。図 7-21 をもう一度見ると，この電圧は非導通トランジスタのエミッタ電圧にもなっていることがわかる。入力トランジスタ Q_1 を ON にするには，入力は V_{E2} を V_{BE} だけ越えなければならない。ゆえに，シュミットトリガが Q_1 を ON にして Q_2 を OFF にする再生的な動作を始める電圧水準（閾値）は，簡単な解析のためには次式で表わされる。

$$\frac{R_3}{R_1+R_2+R_3}V$$

この近似値では，トランジスタの負荷抵抗 R_e の R_3 への負荷効果やトランジスタ間の $(V_{BE})_{on}$ の違いなどは小さい影響しか与えないとして無視してある。これらはよい技術的設計をするのに重要なものであるが，しかし基礎的な回路の理解には必要ではない。

今度は ON にする再生作用について考える。入力信号が閾値を $(V_{BE})_{on}$ だけ越えているときは，トランジスタ Q_1 は導通状態になり始め，V_{C1} の電圧は低くなっていく。トランジスタ Q_2 が OFF になると，V_{E2} はさらに低い電圧の方向へ進む。その結果 Q_1 は ON になる。Q_1 が ON になると，V_{C1} の電圧は下がり，Q_2 のベース電圧を低くして，Q_2 入力駆動電流を少なくして，ついには Q_2 は OFF になる。この再生作用は，Q_1 が飽和状態にあり，Q_2 がカットオフになっている限り持続するであろう。

Q_1 を OFF にし，Q_2 を ON にするときの動作を，図 7-24 に示した部分的な回路図の助けをかりて解析しよう。Q_2 のベース電圧は Q_2 を OFF にしておくのに十分低くなっているはずである。V_{E2} は V_{E1} と同じ電圧であるから，飽和トランジスタ Q_1 の付加電圧すなわちコレクタ-エミッタ間電圧 V_{CE1} が R_3 と R_2 で分割減衰されて，Q_2 のベースに入るときは $(V_{BE})_{on}$ より小さくなっていなければならない。一般的に $(V_{CE})_{sat}$ は $(V_{BE})_{on}$ より小さい。1/5 とか 1/10 位の減衰は確かに Q_2 を OFF にするであろう。減衰率が大きいほど R_3 が小さく，この結果，Q_2 が ON になるときのベース駆動電流から分岐して R_3 に流れる電流がますます多くなることになる。Q_2 を OFF にするときの閾値電圧は，V_{E1} を計算すればわかる（V_{E1} は $I_{E1}R_e$ である）。トランジスタが飽和しているとき（電流利得が大きいとき $I_{B1}\ll I_{C1}$），I_{E1} は近似的に I_{C1} に等しいので，R_2 と R_3 へ流れる電流と飽和トランジスタ中の電圧降下

分を無視すると，電流 I_{E1} は，$I_{E1} \cong V/(R_1+R_e)$ となり，Q_2 を OFF にする閾値電圧は，次の式で求められる。

$$V_{E1}=\frac{R_e}{R_1+R_e}V$$

この式は，前のように簡単化して R_e と R_1 によって V を分割して導びかれたものである。

入力電圧が V_{E1} より下がると，トランジスタ Q_1 は OFF にかわり始める。R_1 を通る電流は少なくなって V_{C1} は上昇するので，Q_2 のベース電圧（V_{B2}）は上がり，Q_2 は ON になっていく。Q_2 を通る電流は V_{E2} を上げ，Q_1 をさらにカットオフの状態に追いやる。この再生作用は，Q_2 が飽和し，Q_1 が OFF になっている限り持続する。

シュミットトリガの代表的な用途はゆっくりと変化している波形を矩形波に直すことである。正弦波を矩形波にする様子が図 7-22 に示されている。前述のように閾値が両方とも 0 V ならば，出力波形は矩形波になる。実際の回路では，両方の電圧のレベルはまったく同じにはなりえず，Q_1 の ON への切りかえと OFF への切りかえの間の電位の違いはヒステリシス電圧と考えられる。それゆえ，出力波形は矩形波とはならないが，入力信号と同じ周波数を持ち，計算機回路に用いられるような鋭く上昇または下降する波形が得られる。このパルス状の信号の周波数と同じにできるということは，シュミットトリガの出力信号の重要な性質となっている。

シュミット回路のもう1つの一般的な用途は，記憶装置の読出し回路からの雑音のある信号を整形することである。この動作に関する信号を図 7-25 に示す。出力は全然対称形ではないが，閾値電圧を越える入力信号と同数だけのパルスが出ている。このように，この回路を通すと，閾値以下のノイズ信号はすべて捨てられて，受け入れるものだけが鋭い出力となる。終りに，もしエミッタ

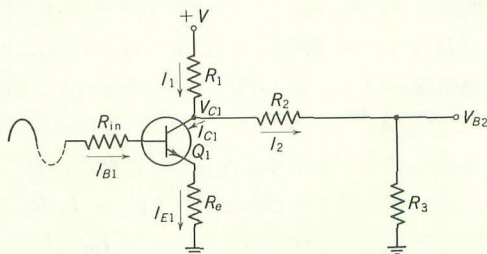



図 7-24 シュミットトリガの部分的な入力回路

抵抗 R_e を接地点よりも低い負の電圧のところに接続すると，閾値水準を 0 V 以下にすることができ，回路の動作はさらに融通性の富んだものになる。

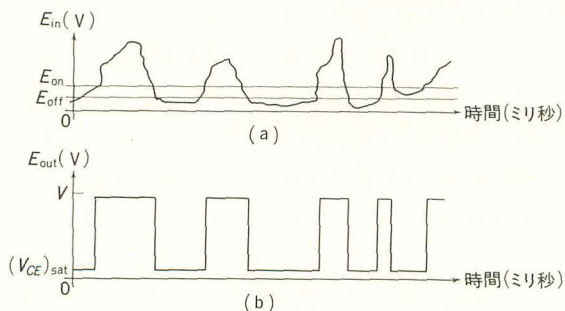



図 7-25　シュミットトリガの整形作用
(a) シュミットトリガへの入力信号　(b) シュミットトリガからの出力信号

問題 7-9　400 Hz の正弦波がシュミットトリガ回路に入ってきたときの出力波形を書きなさい。ただし，$E_{on}=E_{off}=0\text{V}$ とする。時間軸を明記すること。

問題 7-10　図 7-26 に示すような入力がシュミットトリガ回路に入れられたときの出力波形を書きなさい。ただし，$E_{on}=+2\text{V}$, $E_{off}=-1\text{V}$ とする。

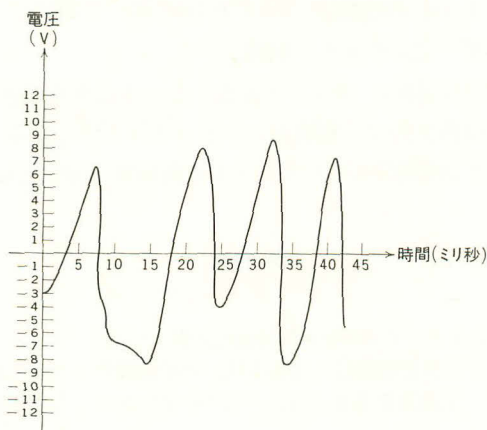



図 7-26　問題 7-10 の波形

要　約

マルチバイブレータ回路は，計算機の動作にとって大切なものである。このうち 2 安定回路は最もよく知られている。動作する場合，回路の状態はいく通りかの方法でかえられる。SET または RESET がくると，回路は決まった

状態に設定される。SET がくると，TRUE 出力は1となり，RESET がくると， FALSE 出力が1となる。フリップ フロップは互いに反対の状態をとるように結線されており，入力パルスがトリガ入力回路に入るたびに状態が反転する。フリップ フロップはまた シフト レジスタ に使うことができる。この動作のためには，方向づけ回路をこの段のトランジスタのコレクタに接続するのではなく， 別の段のコレクタに接続する。 シフト パルス がトリガ入力回路に入れられると，他段に入っていたデータが，この段に移動されてくる。カウンタや，シフト レジスタ としての フリップ フロップの応用は次章で述べることにする。

1安定マルチバイブレータ （ワンショット マルチ） はただ 1 つの安定状態を持ち，一定の長さのパルスを作るのに使われる。パルス時間（幅）あるいは遅れ時間は時定数 RC によって決まる。簡単な設計で，数マイクロ秒ないし数秒の遅れを持たせることができる。一般に，遅れ時間は，$T=0.7RC$ から算出できる。

無安定マルチバイブレータ（刻時装置）は，時定数 RC で決まる一定の周波数を持つ発振器である。周波数は，$f=1/1.4RC$ Hz で算出できる。 上に与えた2つの式は前の章で述べたような制限がある。

シュミット トリガ 回路は，ゆっくり変化する入力波形を“矩形波に直す”のに使われる。回路は再生的で，短時間にスイッチングするようにできている。回路の設計によって，回路がスイッチングする閾値電圧を決めることができる。

問　題

1. PNP 2 安定マルチバイブレータの回路図を書きなさい。

2. PNP トランジスタを使用して 200 kHz の矩形波でトリガされる相補的フリップフロップ のコレクタの波形を書きなさい。同じ時間目盛で入力波形を書き，時間と電圧の目盛を明記しなさい。

3. PNP トランジスタの1安定マルチバイブレータの回路図を書き，回路の時定数が 20 ミリ秒であるような R と C の値を求めなさい。

4. 10 kHz の矩形波入力信号に対して，図 7-27 に示した回路の TRUE の出力波形を書きなさい。同じ時間軸に入力波形も書きなさい。

5. 1安定回路が 100 Hz の矩形波でトリガされるとき，図 7-13 の回路の FALSE 出力波形を書きなさい。

6. 図 7-28 に示した無安定マルチバイブレータの出力刻時周波数を計算しなさい。

7. R と C を 150 kHz の刻時間周波数になるように決めて，PNP の無安定マルチ

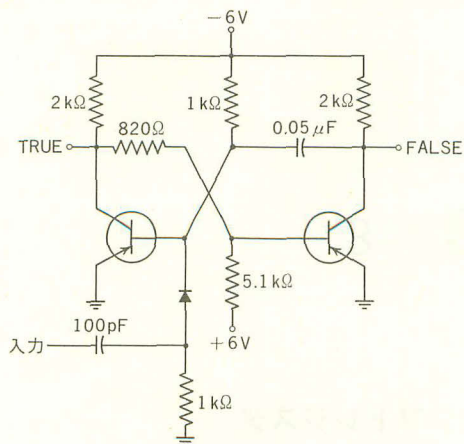



図 7-27　問題4の1安定回路

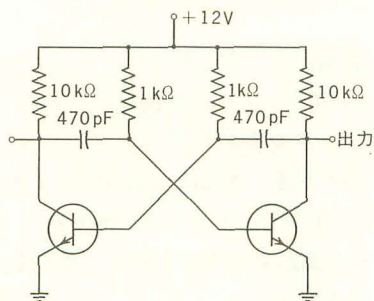



図 7-28　問題6の無安定マルチバイブレータ回路

バイブレータの回路を書きなさい。

8.　図 7-29 に示してある入力に対し，$E_{\mathrm{on}}=+6\mathrm{V}$，$E_{\mathrm{off}}=+1\mathrm{V}$ であるようなシュミット トリガ回路の出力波形を書きなさい。

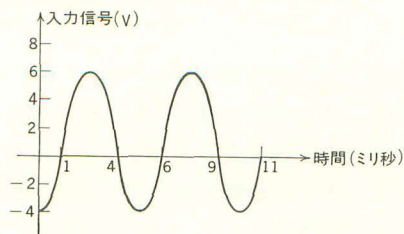



図 7-29　問題8の波形

1000=8

計数器とシフトレジスタ

1安定回路は電子計算機に広く用いられている。また2安定マルチバイブレータは，2進計数器において，いろいろな演算のためのタイミングをとるため，また 計算機の中で2進データを動かす シフト レジスタ として使われている。2進以外で計数する必要のあるときには，基本的な2進計数器を修正して，フィードバック カウンタ にすることができる。無安定マルチバイブレータは，計数器と シフト レジスタ を動かす刻時信号を出すのに使われる。 1安定マルチバイブレータは，短すぎる信号を長くしたり，また長すぎる信号を短くしたり，また1安定回路にトリガ入力があってから一定の時間だけ遅らせて信号を出したりするために使われる。

8-1 2 進計数器

フリップフロップのブロック図を図 8-1 に示す。1 と 0 の出力は各トランジスタのコレクタ出力で，1すなわち TRUE を高い電圧と低い電圧のいずれに対応させるかは任意である。一度決めてしまうと，SET 入力とは TRUE 側を論理的に1の状態にするものと解釈する。トリガすなわち状態を逆にする入力については7章で述べてある。

計数器として使うときには，トリガ入力を1つ前の段の出力からとって，こ

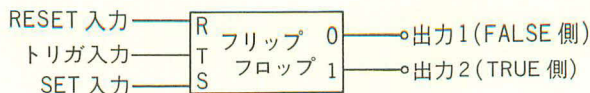



図 8-1　フリップフロップのブロック図

のフリップフロップを次々につないでいく。4段の2進計数器を 図 8-2 に示す。入力信号は 2^0 の段に加えられる。各段にはその次数によって 2^0, 2^1 など

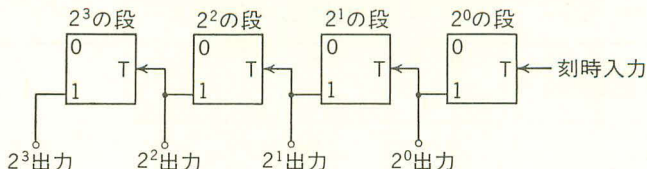



図 8-2　2 進計数器のブロック図

の数が与えられている。各段の出力はフリップフロップの TRUE 側からとられ，各段のトリガ入力も前段の TRUE 側からとられている。刻時入力が負になるたびに，2^0 段は逆の状態になる。すべての段が RESET の状態 (TRUE 出力がすべて 0) であるとして考えてみると，最初の入力パルスは 2^0 出力を 0 から 1 にかえる。いままで NPN 回路を考えているので，入力電圧に負方向の階段があるときにだけ，状態が逆になる。正の論理 ($+V$V$\equiv$1, 0 V$\equiv$0) のときには，入力が 1 から 0 ($+V$V から 0 V) になるときフリップフロップは逆になることがわかる。最初の入力トリガパルスが，2^0 段の出力を 0 から 1 にしたので，2^1 段は変化しない。2^0 段の状態だけが変化した。第 2 の入力パルスで，2^0 段は再び逆になる。このとき出力は，1 から 0 になる。これは 2^1 段を逆にするので，2^1 段の出力は 0 から 1 になる。それ以上の段には，トリガパルスは入らない。このような状態変化を 表 8-1 に示す。

計数器に示される 2 進数は，入力パルスの個数と同じである。最初のパルスで 0001 (右端が 2^0 すなわち最小桁 LSD)，第 2 で 0010，第 3 で 0011，第 4 で 0100，… となり，15 すなわち 2 進数 1111 まで進む。その次のパルスで数は 2 進の 0000 に戻り，再び計数が始まる。フリップフロップが 4 段のときには，16 個のパルスごとに計数を繰り返すが，一般に n 段計数器では 2^n 個の計数ができる。

わき道にそれるが，表 8-1 の矢印は，各段で，1 から 0 にかわるとき，トリガ入力が次段へ入ることを示している。2^0 段は，8 サイクル変化し，2^1 段は 4 サイクル，2^2 は 2 サイクル，2^3 は 1 サイクルである。この事実は，高い段で変化率が少ないということである。16 個のパルスに対して，第 1 段は 8 回

(16/2¹), 次は4回 (16/2²), 第3段は2回 (16/2³), そして第4段は1回 (16/2⁴) である。この変化率の減少する様子は, 図 8-3 のようなタイミングダイアグラムで表わすこともできる。図には, 入力の刻時パルスと, 各段の TRUE 出力を示してあって, 各段の周波数は 1/2 ずつ少なくなっていることがわかる。そこで, 図 8-2 に示したものは 周波数分割器でもある。たとえば, もし

表 8-1 4段計数器の状態変化

入力パルス	2^3 出力	2^2 出力	2^1 出力	2^0 出力
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16 または 0	0	0	0	0

入力周波数が 256,000 Hz なら, 2^3 段の出力は 16,000 Hz (256 kHz/16) である。図 8-3 の矢印は, 次の段にトリガ入力を加える負の傾斜を示している。

計数することに話を戻すと, 図 8-2 の計数器は 漸増計数 (count up) レジスタである。このレジスタは2進のアキュムレータまたは加算器として使える。もし5個のパルスが入力に加えられると, カウントは 5 すなわち 0101 (2 進表示したもの) になる。もしさらに4個のパルスが入ると, 全体で 1001 となる。このように, 計数器を, パルスの数を加えるためのアキュムレータとしてカウンタを使って, 2つの数の加算を行なうことができる。減算を行なうためには, レジスタは逆方向に計数, すなわち漸減計数 (count down) しなければならない。この機能を持たせるためには, 図 8-4 に示す結合を用いる。漸減

計数動作のためには，トリガパルスは前の段の FALSE 側からとる。しかし

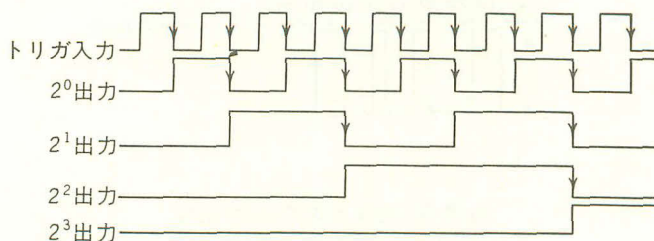



図 8-3　周波数分割器の波形

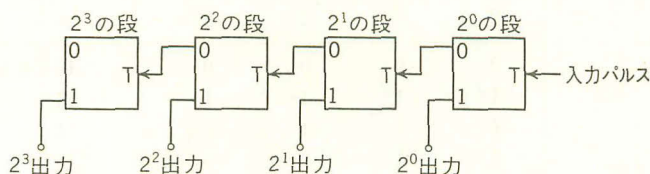



図 8-4　漸減計数レジスタ

ながら，TRUE 側を出力として使うので，FALSE 側からとっているトリガパルスが1から0になるとき次の段がかわるが，TRUE 出力を示す表の上では，0 から1になるとき次の段にトリガ入力が加えられることになる。表 8-2は漸減計数動作を示す。

表8-2 の矢印も，より高い段がより低い率で働くことを表わしている（漸増計数または漸減計数レジスタのいずれの接続でも，周波数分割器としても動作する)。算術演算としては，一方の接続法（TRUE の側をトリガ出力として使う）は加算になり，他方は減算となる。たいていの大形汎用電子計算機では，普通の2進加算器を用いるが，特殊用途の計算機にはここで説明したようにアキュムレータとしてカウンタを使うものもある。次にあげた問題は，2 安定マルチバイブレータ回路を用いるとき，考慮しなければならない重要な特徴を認識するのに役立つであろう。

問題 8-1　PNP 2安定マルチバイブレータの回路を書きなさい。

問題 8-2　PNP フリップフロップは電圧変化が正の方向のとき変化する。図 8-5 のトリガ信号が入ったときの，フリップフロップの FALSE (0) 側の出力を書きなさい。

問題 8-3　5段の漸減計数レジスタのブロック図を書きなさい。

問題 8-4　PNP のフリップフロップを用いた5段レジスタについて，漸減計数の表を書きなさい。

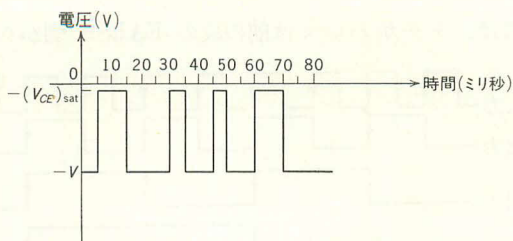
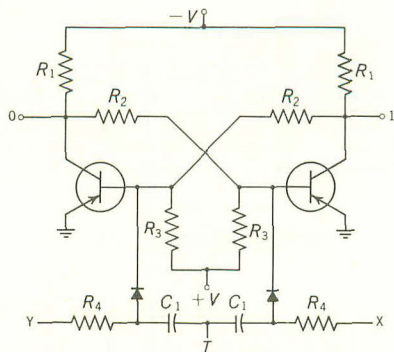



図 8-5　問題 8-2 のトリガ信号

表 8-2　漸減計数動作（4段）

入力パルス	TRUE 側 2^3	2^2	2^1	2^0	10進出力 カウント
0	0	0	0	0	16（または 0）
1	1	1	1	1	15
2	1	1	1	0	14
3	1	1	0	1	13
4	1	1	0	0	12
5	1	0	1	1	11
6	1	0	1	0	10
7	1	0	0	1	9
8	1	0	0	0	8
9	0	1	1	1	7
10	0	1	1	0	6
11	0	1	0	1	5
12	0	1	0	0	4
13	0	0	1	1	3
14	0	0	1	0	2
15	0	0	0	1	1
16（または 0）	0	0	0	0	0（または16）

8-2　シフト レジスタ

前章では，フリップ フロップ の方向づけ（steering）回路について考えたが，それは導通トランジスタにだけ OFF のパルスを加えるものであった。その回路では，方向づけの抵抗が フリップ フロップ トランジスタ のコレクタに接続されていた。他の可能性として，方向づけの抵抗を他の段の フリップ フロップに結ぶことが考えられる（図8-6 の回路）。そうすると，トリガ パルスが入ったとき，結んだ段の状態でこの段がどうかわるかが決定される。配線のし

(a) フリップフロップ回路（シフト用方向づけ回路付）

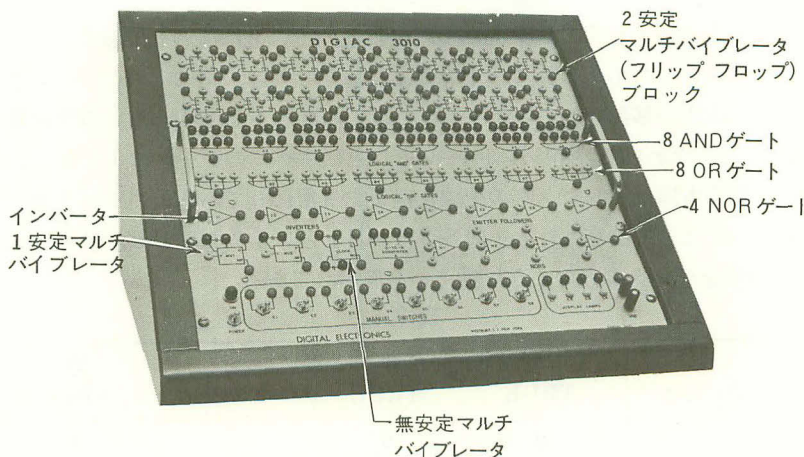
(b) フリップ フロップ のブロック

(c) 論理操作の研究のための計算機ブロック (Digital Electronics, Westbury, N.Y. の好意による)。

図 8-6

かたによって，格納されている情報（2 進の 1 または 0）はその段から他の段（右または左）に移ることになる。このように，配線によって，この回路は右シフト レジスタ または 左シフト レジスタになる。図 8-7, 図 8-8 に例を示す。

ブロック図では，方向づけ抵抗を接続する点XとYで表わしてある。図 8-6 に，表示法と対応する回路の点を示してある。X入力はいつも FALSE 側

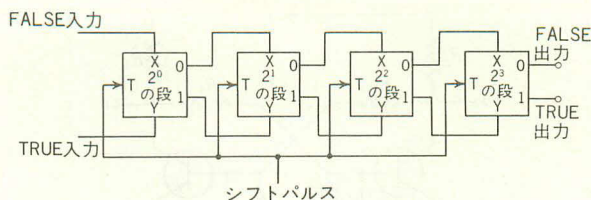



図 8-7　右シフトレジスタ

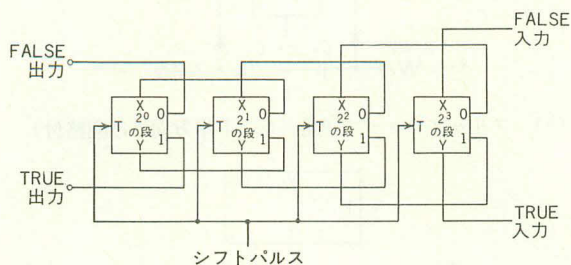



図 8-8　左シフトレジスタ

に，Y入力はいつも TRUE 側につながれていることに注意しておく。

レジスタの一方から入れられた情報またはデータは，各シフトパルスごとに1ビットずつ移動される。データは一度に1ビットしか動かないので演算は“直列”に行なわれる。各シフトパルスごとにすべてのデータは1ビットだけ動く。動く方向は，レジスタの配線を左シフト用にしてあるか右シフト用にしてあるかによって決定される。表 8-3 は 0110 という入力に対し，シフトレジスタの中をデータがどのように動くかを例示してある。

表 8-3

入力パルス	2^0 TRUE	2^1 TRUE	2^2 TRUE	2^3 TRUE
	0	0	0	0
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	0	1	1	0

データは4つのパルスを受けて，レジスタの中に移された。語を正しく扱うためには，使うパルスの数はレジスタの段数と等しくなくてはならない。情報をレジスタに移すということの他に，しばしば情報をレジスタの中に保持することが必要になる。これは，データを再び入力にまわすことによって行なわれる。循環桁移動のできる3段レジスタの回路を 図 8-9 に示す。この論理回路を

用いて，入力用桁移動あるいは循環桁移動の操作を行なうことができる。各シフト操作ごとに，シフト パルス 入力線に 3 個の シフト パルス が加えられる。もしデータの循環桁移動入力に信号があれば（そして入力用桁移動信号がなければ)，レジスタ初段 (2^0) に入るデータは 2^2 段からのものである。3 回の シフト パルス のあとには，シフト操作の前と同じ語がレジスタに入っている。このように，データは循環桁移動され，蓄えられる。新しいデータをレジスタに入れるときは，循環桁移動は止めておかなければならない。この場合，データ入力用桁移動信号がある（そして循環桁移動信号はない）ので，新しい入力データが入力段に加えられる。3 回の シフト パルス のあと，新しいデータがレジスタに蓄えられる。循環桁移動と入力用桁移動とを同時に指令してはならない。

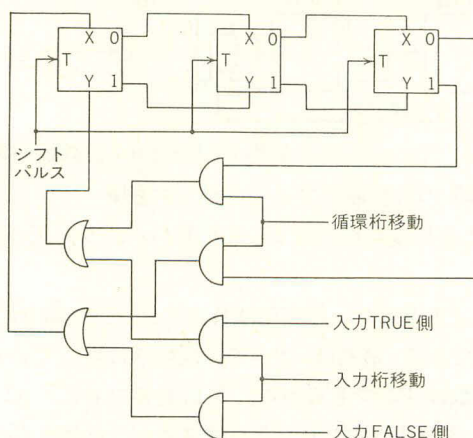



図 8-9 入力用桁移動および循環桁移動の操作

8-3 フィードバック計数器

いままで考えてきた計数器は，2 進数としてしか数えることができないが，それ以外のものが必要なときがある。最も一般的なのは，10 進計数のため 10 を基数として数える方法である。2 進計数を他の方法にかえるいろいろな技法があるが，よく使われている方法の 1 つに，フィードバックを使って計数する方法がある。ある数え方が必要なとき，その基数より大きいすぐ次の 2 進計数ができるように段数を選び，余分の数だけカウントを進めるために，フィードバックによるカウントを使うのである。たとえば，6 を計数する場合，3 段計数器（8 を計数できる）と，計数器を 2 ステップ進めることのできるフィードバ

ック回路を使えばよい。8 引く 2 で必要な 6 のカウントができる。3 段の 2 進計数器は 8 個のパルスのあとに零に戻り，再び計数し始めるのと同様に，大きさ 6 の計数器は 6 ステップごとに零に戻り，計数を繰り返す。図 8-10 に回路の 1 例を示す。大きさ 6 の計数器は “6 を法” (MOD 6 と書く) とする計数器と呼ぶことがある。法 (module) の数だけのパルスが入ると計数器は零に戻る。指定された電圧変化（正または負）がトリガ入力となるから，設計の際考慮する必要がある。正の論理を使うとき，1 は 0 より高い電圧であるから，1 から 0 になることは負の電圧変化である。負の論理では，1 は 0 より低い電圧であるから，1 から 0 になることは正の電圧変化である。

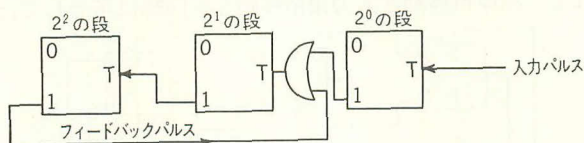



図 8-10　フィードバックを用いた大きさ 6 の計数器（概念図）

例　8-1　MOD 25 の計数器をフィードバックを使って作りなさい。フリップフロップは負にかわるパルスによって状態がかわり，正の論理を使うものとする。

解：正の論理では 1 から 0 への変化は負のステップであり，0 から 1 への変化は正のステップである。最初は，すべての段は零になっていると仮定する。25 のすぐ上の 2 進計数は 32 であるから，5 段必要である。32 を 25 に減らすためにステップは 32−25，すなわち 7 だけ進めなければならない。7 進めるためには，$1+2+4=7$ だから，$2^0(1)$，$2^1(2)$，$2^2(4)$ の段にトリガパルスを加えればよい。このための回路を 図 8-11 に示す。この例ではフィードバックを 0 出力からとっている。2^4 の段は最初 RESET の状態にあり，0 すなわち FALSE 出力は 1 である。$16(2^4)$ 個のパルスが入ると，その 2^4 段の FALSE 出力は 1 から 0 にかわり，1 安定マルチバイブレータに負のパルスをトリガ入力として加える。トリガ入力が加えられると，1 安定マルチバイブレータの 1 の出力は 2^0, 2^1, 2^2 段を SET するパルスを出し，カウントを進める。この際，どの段の状態が変化しても，他の段にトリガ入力が加わらないということは重要なことである。この例では，最初の 3 つの段は 0 から 1 になる。正のステップはトリガ入力とはならない。表 8-4 は各入力パルスに対するカウントを表わしたものである。

フィードバックは 1 サイクルに 1 回しか変化しない最後の段からとってい

る。例8-1のフィードバックを2^3段出力からとったとすると，その段はサイクルごとに2回変化するため，$2\times7=14$だけカウントを進めてしまう。これは設計上注意すべき事項であるが，別のカウントの進め方があることを示している。カウントを2の倍数だけ進めたいときには，フィードバックはより低い段からとることができる。例 8-2 にこの方法を示す。

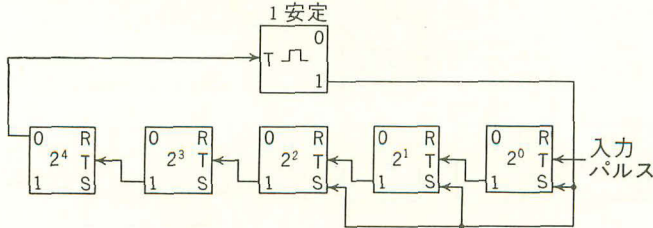



図 8-11 例 8-1 の MOD 25 の計数器

表 8-4 MOD 25 の計数器のステップ

入 力 パルス	2^4 TRUE	2^3 TRUE	2^2 TRUE	2^1 TRUE	2^0 TRUE	
0	0	0	0	0	0	
1	0	0	0	0	1	
2	0	0	0	1	0	
-	-	-	-	-	-	
-	-	-	-	-	-	
7	0	0	1	1	1	
8	0	1	0	0	0	
-	-	-	-	-	-	
-	-	-	-	-	-	
-	-	-	-	-	-	
15	0	1	1	1	1	
16	1	0	0	0	0	2^4段のFALSE出力は1から0にかわり$2^0, 2^1, 2^2$段をSETする。
	1	0	1	1	1	
17	1	1	0	0	0	
18	1	1	0	0	1	
-	-	-	-	-	-	
-	-	-	-	-	-	
24	1	1	1	1	1	
25 または 0	0	0	0	0	0	

例 8-2 MOD 26 の計数器の回路を作り，そのカウント表を書きなさい。正の論理と負のトリガパルスを使うものとする。

解：MOD 26 の計数器を作るためには，6 ステップ進めるフィードバック回路のついた 5 段計数器（最大 32 カウント）が必要である。6 進めるために，2^4 段の FALSE 出力を 2^1 と 2^2 段にフィードバックさせればよいが，2^3 段から 2^0 と 2^1 にフィードバックさせてもよい。なぜなら，この場合，カウントを 3 ずつ 2 回進めることになるからである。カウント表とブロック図を表 8-

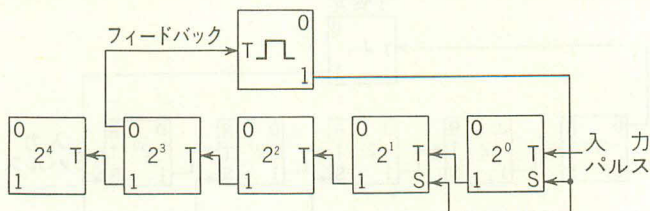



図 8-12　例 8-2 MOD 26 の計数器

表 8-5　例 8-2 のカウント表

入　力 パルス	2^4 TRUE	2^3 TRUE	2^2 TRUE	2^1 TRUE	2^0 TRUE	
0	0	0	0	0	0	
1	0	0	0	0	1	
2	0	0	0	1	0	
–	–	–	–	–	–	
–	–	–	–	–	–	
7	0	0 ┐	1	1	1	
8	0	1 ←	0	0	0	カウントが 3 進められる。
	0	1	0	1	1	
9	0	1	1	0	0	
–	–	–	–	–	–	
–	–	–	–	–	–	
–	–	–	–	–	–	
–	–	–	–	–	–	
20	1	0 ┐	1	1	1	
21	1	1 ←	0	0	0	カウントが 3 進められる。
	1	1	0	1	1	
22	1	1	1	0	0	
23	1	1	1	0	1	
24	1	1	1	1	0	
25	1	1	1	1	1	
26 または 0	0	0	0	0	0	

↑

矢印は 2^3 段が 1 安定マルチバイブレータにトリガ入力を加えることを示す。

5 と図 8-12 に示す。

問題 8-5 MOD 12 の計数器のブロック図と表を書きなさい。フリップフロップは PNP (トリガ入力は正の方向パルス) で，負の論理を使うものとする。

問題 8-6 MOD 19 の計数器のブロック図を書きなさい。NPN フリップフロップを用い，負の論理を使うものとする。

問題 8-7 問題 8-6 を MOD 39 の計数器について解きなさい。

問題 8-8 問題 8-7 のカウント表を作りなさい。

問題 8-9 MOD 24 の計数器について，フィードバックを使って 2 つの異なったブロック図を書きなさい。トリガの極性と使っている論理の型を示しておくこと。

10 進の計数器もフィードバックを使って作ることができる。10 進計数器は一般的に用いられるものなので，ここでは別に取り扱うことにする。これまで述べてきたように，10 のカウントは 4 段計数器を用い，6 進めることにより作ることができる。6 進めるには，2^1 と 2^2 段を 1 回トリガするか，2^0 と 2^1 段を 2 回トリガすればよい。この各方法とカウント表を，図 8-13 と図 8-14，および 表 8-6 と表 8-7 に示す。

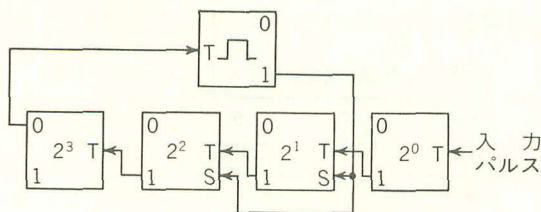



図 8-13 6 進めるパルスを 1 回用いた10進計数器

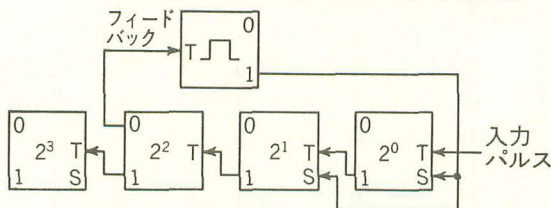



図 8-14 1 サイクルあたり 3 進めるパルスを 2 回用いた10進計数器

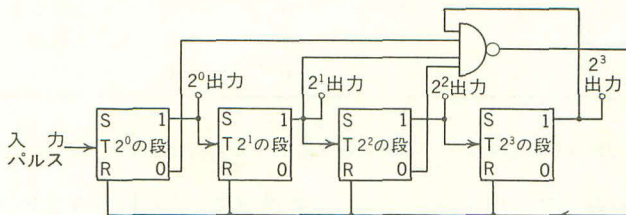



図 8-15 2 進カウントにそった10進計数器

表 8-6　10進計数器（図 8-13）のカウント表

入　力 パルス	2^3 TRUE	2^2 TRUE	2^1 TRUE	2^0 TRUE	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0 ┐	1	1	1	
8	1 ←	0	0	0	} カウントが6進められる
	1	1	1	0	
9	1	1	1	1	
10（または0）	0	0	0	0	
	↑				

矢印は 2^3 段が1安定マルチバイブレータにトリガ入力を加えることを示す。

表 8-7　10進計数器（図 8-14）のカウント表

入　力 パルス	2^3 TRUE	2^2 TRUE	2^1 TRUE	2^0 TRUE	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0 ┐	1	1	
4	0	1 ←	0	0	} カウントが3進められる
	0	1	1	1	
5	1	0	0	0	
6	1	0	0	1	
7	1	0	1	0	
8	1	0 ┐	1	1	
9	1	1 ←	0	0	} カウントが3進められる
	1	1	1	1	
10（または0）	0	0	0	0	
		↑			

矢印は 2^2 段が1安定マルチバイブレータにトリガ入力を加えることを示す。

10 進計数器では，10 パルスごとに，2 進カウントと同じカウントにしなければならないことがよくある。その場合，計数器を零にするため，最終のカウ

表 8-8 10進計数器（図 8-15）のカウント表

入 力 パルス	2^3 TRUE	2^2 TRUE	2^1 TRUE	2^0 TRUE
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
↓	↓	↓	↓	↓
0	0	0	0	0

ント（この場合 10）を検出するのに**解読ゲート**というものを使う。この種の回路を図8-15に示し，そのカウント表を表8-8に与える。カウントが 1010（2進の 10）になったとき，NAND ゲートの出力は負方向に変化し，4 つの段をリセットする。表によれば，2^1 と 2^3 段のフリップフロップを零にすればよいことがわかる。すべての段はいつでも零からカウントがスタートするようにリセットされるが，これは実は必要でなかった。表によれば，各 10 進ステップでの2進カウントと 10 進カウントとはまったく対応しているが，フィードバックカウンタではそうはなっていない。もとに戻って，このことが事実であることを調べなさい。2 進-10 進変換器を使うと，各ステップで 10 進カウントを得ることができる。この型の回路は，カウントを 10 進数として表わす表示灯をつけるために使うこともできる。数個の 10 進計数器をつないで，10 進法による計数をするのに使うこともある（図 8-16）。

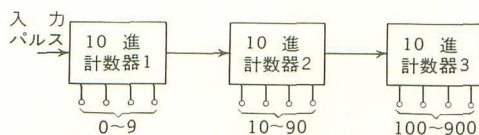



図 8-16 10 進法による計数器

要　約

電子計算機の基本的な部分は計数器である。相補的なフリップフロップをいくつか接続して，漸増計数または漸減計数の操作ができる。n 段では，カウントの数は 2^n である。たとえば，3段では8カウント（2^3），5段では32カウント（2^5）である。

2進カウント以外のものが必要なとき，カウントを進めるために，計数フィードバック回路が使われる。たとえば，MOD 28 の計数器では，5段計数器（$2^5=32$）を用い，28 カウントごとにサイクルを繰り返すため，途中でカウントを $32-28=4$ だけ進める。2安定素子の他の応用例として，シフトレジスタがある。1つのレジスタに蓄えられたデータは，直列の演算を行なうために他にシフトされることもある。10 章で演算装置について述べているが，そこでシフトレジスタの若干の応用例を扱う。

問　題

1. 5段漸減計数レジスタのブロック図を書きなさい。
2. 4段左シフトレジスタのブロック図を書きなさい。
3. MOD 20 の計数器のブロック図を書きなさい。
4. MOD 12 の計数器のブロック図を書きなさい。
5. MOD 12 の計数器のカウント表を作りなさい。
6. 2つの異なる MOD 3 の計数器のブロック図を書きなさい。
7. 10 進計数器のブロック図を書きなさい（この章にあるものと違うもの）。
8. MOD 24 の計数器のブロック図を書きなさい。

第 III 部　計算機装置

1001 ═ 9

電子計算機のタイミングと制御

電子計算機は考えたり，自動的に動いたりしているように見えるが，実は，たいへん簡単なことを順番に実行しているにすぎない。制御装置は，計算機の一部であって，基本的な刻時信号を出して，計算機のすべての装置を動かしている。記憶されているプログラムは，実行すべき動作の主な流れを指示するものであるが，制御装置は，実行されるべき個々の命令に対して，それぞれがとらなければならないすべての小さな操作を準備するものである。

“配線”による動作と記憶されたプログラムによる動作の違いを正しく認識しておく必要がある。“配線”による動作というのは，計算機の中の特別な回路によって自動的に実行される動作のことである。たとえば，乗算や除算に対しては，これらの1つの命令が出されると動作を開始し，この命令を実行するのに必要なすべての小さなステップが制御装置によって実行されるように結線してあることが多い。計算機に加減算の命令しかなくても，乗除算は多くのステップ数を持つ記憶されたプログラムで行なうことができる。両者の違いは，主として実行の速さの点にあり，配線による動作の方が速い。

制御装置は，計算機ごとにかなり異なる。これはどのような命令が組み込まれているかとか，内部記憶装置の型とか，あるいは動作が直列的か並列的かなどによって左右される。制御装置の詳細について述べるのは複雑であるが，まず，基本的な部分から学んでいこう。基本的なものとしては計数器，ダイオード マトリックス，解読器，およびレジスタなどがある。すべての動作が主刻時信号（マスタ クロック，たとえば無安定発振器）で制御されているときには，その計算機は同期的に動作するという。同期演算が遅すぎる（次のステップを始めるのに，一定の時間待っていることが必要である）ときには，非同期的演算が使われ，現在のステップが完了したら直ちに次のステップを開始するようにする。

制御装置の全体としての働きを考える前に，いくつかの基本的な部分を研究していくことにしよう。

9-1 刻時信号

ある特別な計数を 図8-15 の NAND ゲートで行なったように，ある時間間隔は計数器の出力を解読することによって得ることができる。2 段の計数器は $2^2(4)$ 個の異なった状態を持つことができる。この各々は，それぞれ違う時間間隔で生じる。各組合せを順番にとらえることによって，計算機で使うためのいくつかのパルス系列を得ることができる。図9-1 は2段の計数器とそのための**解読ゲート**である。図 9-2 はゲートから出てきた**刻時信号**の波形である。図 9-2 を見ると，出力 0,1,2,3 はその番号順に並んでいるという重要な特徴がある。どの時点でも ON になっているのは 1つしかないから，各ステップに固有の順番を与えることによって刻時信号としてこれを使うことができる。たとえば，2 つの数を加え合わせるには，第0ステップで加算レジスタを払い，第1ステップで第1の数をAレジスタへ加え，第2ステップで第1の数へ2番目の数を加え，その結果を加算レジスタに入れる。第3ステップでこれを読んで記憶装置にしまう。言いかえると，計算機が行なう小ステップはこのようなタイミングと解読ゲートによって制御されているということになる。計算機のプログラムは，加算，減算などをどんな順にどのデータに対して行なうかということを示すだけである。それぞれの小さな動作に対して，計算機は決まった型のステップの組合せを実行する。ステップが2のべき乗個でない場合には，フィードバック計数器を使って，1 サイクルにつき必要なだけのステップ

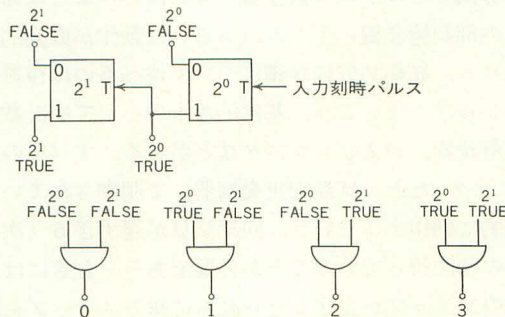



図 9-1　2段計数器と解読ゲート

数を与える。この方法はステップが少ない場合によい方法なのであるが，たくさんのステップを要する場合には，かなりの数の解読ゲートが必要となる。この例では，4 つのステップに対し，2 段計数器で 4 つのゲートが必要であった。64 個のステップがあれば，64 個のゲートを持った 6 段計数器が必要となり，各ゲートはそれぞれ 6 個の入力を持たなければならない。ダイオード入力では,解読ゲートのためにダイオードを 64×6=384 個使わなければならない。この数を減らすのに，ダイオード マトリックス を使う方法があるが，これについてはあとで述べることにする。

タイミングのためのパルスを順次得るもう 1 つの方法は，**リング カウンタ**の出力を使うものである。リング カウンタ は，任意の時刻に TRUE 出力が 1

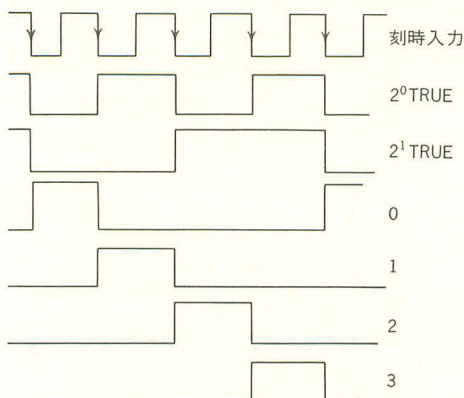



図 9-2 図 9-1 に対する波形

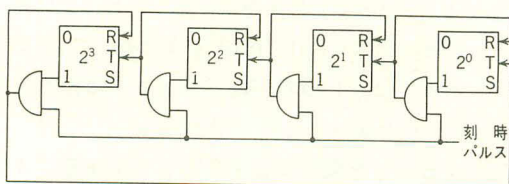



図 9-3 リングカウンタ

つしかないという点で他の計数器と異なっている。実際，フリップ フロップの配列は シフト レジスタ に似ているが，この回路は慣例上リング カウンタと呼ばれている。4 段の リング カウンタ を 図9-3 に示す。

最初に 1 つの段だけが 1 となっているとすると，刻時パルスが入ることにより，次の段は 1 になり，前の段は 0 にリセットされる。リング カウンタ の出

力波形（図 9-4）を見ると，やはり刻時信号を順次出していることがわかる。リング カウンタ は，4 つの フリップ フロップ を使っているが，解読ゲートがなくても，連続的な刻時信号を出している。

もう 1 つの リング カウンタ は，シフト レジスタ を使って，1 を循環桁移動していくものである。図 9-5 にそのブロック図を示す。1 を 1 つだけレジスタの中に入れておき，刻時パルスをトリガ入力として入れると，この 1 は 1 段ず

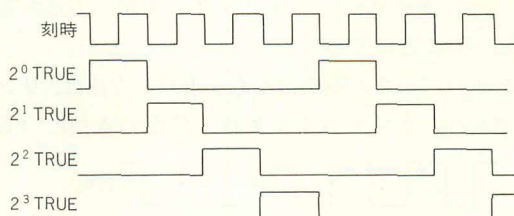



図 9-4　リングカウンタの出力波形

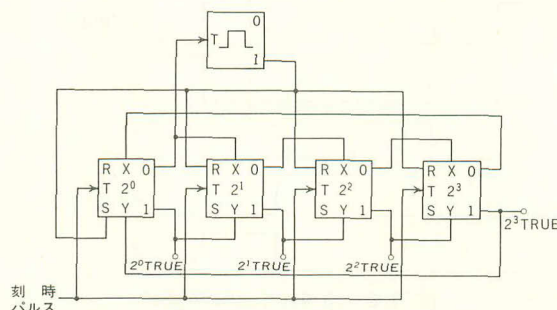



図 9-5　シフト レジスタ を用いた リング カウンタ

つ進んでいく。それぞれの TRUE 出力が次々 1 になって，2^0 から 2^3 までひとまわりする基本サイクルは，4 つの刻時パルスごとに繰り返されることになる。出力の波形は 図 9-4 とまったく同じである。

1 が 1 つだけ存在することを保証する手段として 1 安定マルチバイブレータを用いる。2^0 の段が 1 になるとき，そこに トリガが入って 2^0 段を SET し，他の段を RESET することになる。例えば，2 番目の フリップ フロップを 1 にするような雑音（ノイズ）パルスが現われたとしても，サイクルの間には計数器としての正しい動作に戻るであろう。正しい動作のためには 2^0 が 1 になったときに，すべての他の段は 0 である必要があるから，1 安定回路は他のものを変化させてはならない。ここでは，1 安定回路はすべての段でただ 1 つの 1 が現われるように，修正用にだけ使われていることに注意しよう。もしどの

段にも1がなかったら，回路はどこにも出力を出さないことになることは明らかであろう。

問題 9-1 5段リング カウンタ のブロック図を書き，1 kHz の刻時信号のもとでの各段の出力波形を示しなさい。

問題 9-2 3段の計数器と，8つの解読ゲートのブロック図を書き，各ゲートに対する入力信号に名前をつけて明示しなさい。

問題 9-3 問題 9-2 の各ゲートについて，刻時入力波形の下にそろえて，出力波形を書きなさい。ただし，NPN フリップ フロップと，正論理が使われているものとする。

9-2 符号化および解読用マトリックス

計算機で使用するために2進数を1個の信号に符号化する別の方法は，**ダイオード マトリックス** である。入力と出力の働きの例をいくつかあげてみよう。図 9-6 は3変数解読器（three variable decoder）用のダイオード マトリックスである。各入力の TRUE と FALSE の両方が与えられているから，いつでも入力の半分は1で半分は0である。入力の可能な組合せ（この場合 2^3 すなわち8個）に対応して，出力が決まる。しかも，いつも1個の出力線だけが高い電圧になっている。そして1つの出力線が高い電圧になっているとき，他のすべての入力の組合せについては，少なくとも1つのダイオード入力が0 Vであって，それらにつながる他の出力線を0 Vにしている。黒く塗ったダイオードは，高い電圧の入力（$+V$）を持つもので，逆バイアスがかかっている

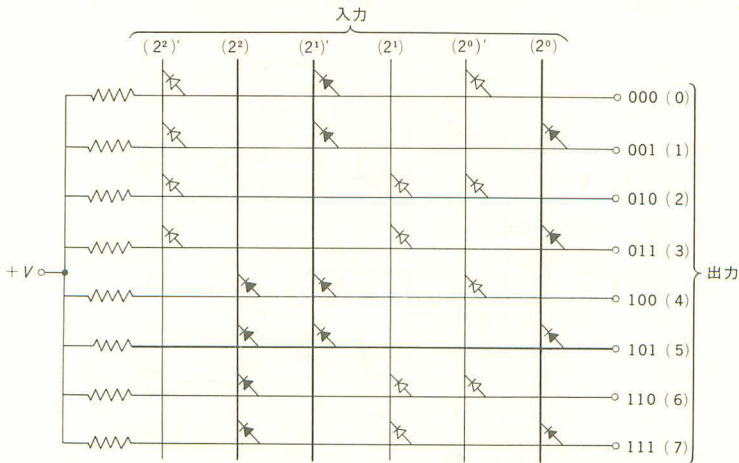



図 9-6 8つの解読用 ダイオード マトリックス（2進数 000-111）

ことになる（101 の組合せのときの様子を示している）。$(2^2)'$ における低い入力電圧（0 V）は，上方の 4 本の出力線を低い電圧に抑える働きをし，(2^1) の低い入力電圧は下の 2 本の出力線を抑え，$(2^0)'$ の低い入力電圧は，100 の出力線に低電圧を与えている。こうして，101 の線にだけ高い電圧が与えられることになる。$2^3\times3$ すなわち24 個のダイオードが必要であることに注意しよう。一般に 2^n 個の出力に対しては $2^n\cdot n$ 個のダイオードが必要である。6 段の計数器（64 の計数をする）では $2^6\cdot6$ すなわち 384 個のダイオードが必要である。

入力を“木”状に配列することにより 8 出力を得るための入力素子の数を 14 個に減らすことができるだろう。ダイオード マトリックス が数学でのマトリックスの行と列の規則正しい配列に類似して名づけられたように，**木状結線** (tree connecting) もその物理的な姿（図 9-7 参照）から名づけられている。

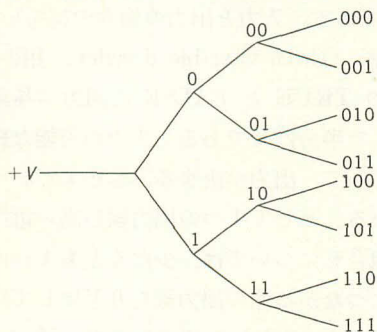



図 9-7　木状結線の図

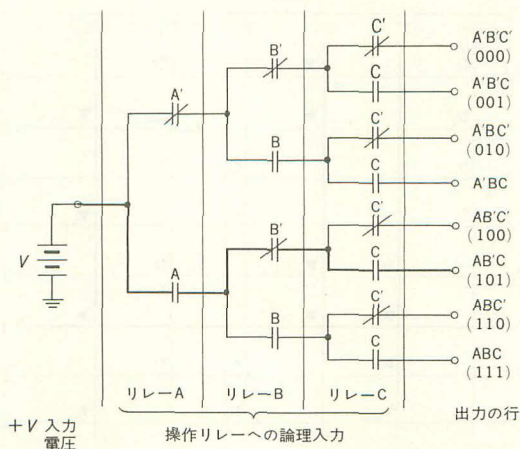



図 9-8　リレー論理回路を用いた 3 変数に対する木状結線

この木状結線の概念を具現するためには，図 9-8 のように，リレーを使った回路を作ればよい（5-2 節に述べたリレー論理回路を参照)。ここでは，3 個のリレーが使われている。リレーAは2つの接点を持ち，うち1つは通常閉じていて（NC)，いま1つは通常開いている（NO)。リレー B はこの組が 2 つ（4 つの接点）あり，リレーCには4組（8つの接点）ある。もし，論理として AB′C という入力に対応して，リレー A と C とを選択すれば（リレーを動作させると)，$+V$ の電圧は，図 9-8 に太線で示してあるように，A の閉じた NO 接点，B の NC 接点，C の閉じた NO 接点を通過して伝わる。

この回路を使うと，選ばれた1つの線は $+V$ V になるが，それ以外の線の出力は0 Vである。この木状結線を使うと，14 個の接点を用いることによって，8 つの出力線の 1 つを選択することができる。この考えを 6 入力に拡張すると，2^6 すなわち 64 個の出力を得るための木状解読器では，

$$2^1+2^2+2^3+2^4+2^5+2^6$$

すなわち，126 個の接点が必要である。一方，ダイオード マトリックス では 6 $\times 2^6=384$ 個の接点が必要である。木状構造は素子の数が少なくてよいようであるが，素子として直接ダイオードを使うことはできない。しかし，全出力信号の一部の解読には，ダイオードの数を減らす目的で，使うことができる。現在のところ，ダイオードがより一般的に用いられている選択素子であるから，ダイオード マトリックス に戻って，さらに述べていくことにする。

ダイオード マトリックス は，2 進符号を解読して 1 つの出力線を得たり，また，多くの出力線をとってきて，それを独得の2進コードに符号化したりするのに使われる。解読の例として，算術演算をする際に3余りコードを用いている計算機システムを考えよう。もし最終結果を 10 進数としてディスプレイに写したいなら，各3余りコード（10 進法の数字として 10 ある）をそれぞれ違った10進法の数字として写し出さなければならない。図 9-9 のマトリックスは 4 段の バッファ（緩衝）レジスタ から信号が送られている。このレジスタは，表示のために解読されている間，2 進数をためておいて，計算機が休まずに，動作を続けられるように設けられたレジスタである。計算機が他の動作をし，あるいは表示する新らしい数を計算しているときでさえ，ディスプレイに写している数は バッファ レジスタ の中に残っている。新らしい値をディスプレイに写すときには，3 余りコードを1ビットずつ桁移動するか，または全ビット一度に移して（ダンプして）バッファ レジスタ の中へ入れる。このシフトの方法は，計算機が直列的に動作しているか並列的に動作しているかによ

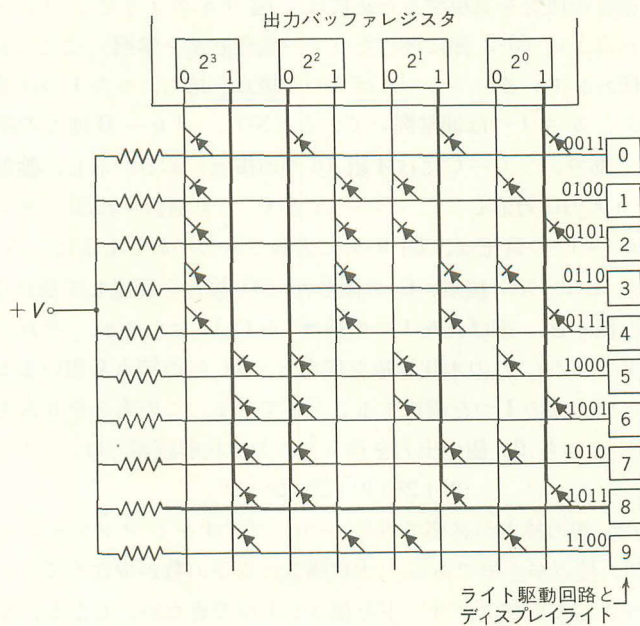



図 9-9　3余り数に対する解読マトリックス

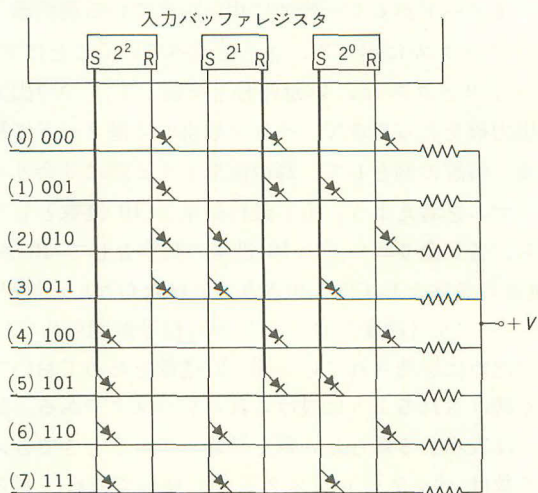



図 9-10　8進数に対する符号化マトリックス

って決まる。

マトリックスへの入力が フリップ フロップ からきているので, TRUE, FA-

LSE 両方の端子が利用できる。この図では，10 進法の 8 の解読を示している。直接的な解読法をとっているので，40 個のダイオードが使われている。マトリックスによって表示すべく選ばれた 10 進法の数字はマトリックスに負荷がかからないように，ライト駆動回路*（トランジスタ増幅器でできている）によって操作されている。

図 9-10 は 8 進数に対する符号化マトリックスである。この図では，0~7 の線のうちの 1 つから 0 V の入力があると，これに対応する 2 進数が作られて，入力バッファ レジスタ の中に入る。太線は 8 進数の 5 が入ってきたときの動作を示している。注意すべきことは図示の供給電圧はすべてのダイオードに逆バイアスとなっているので，これらはレジスタの各段には何の影響も与えないことである。1 つの入力だけが低く（0 V）なると，入力バッファ レジスタ に入っていた内容はかえられる。

9-3 制 御

制御装置の動作は計算機の細部に大きく左右される。すなわち，直列に動作するか，並列に動作するかとか，記憶装置はどんな型か，どんな入出力を使うか，あるいは算術演算はどのように行なわれているかなどによって決まってくるのである。これはまた，計算機がいかにしてプログラムされるかということにも左右される。2 章で用いた機械語は 1 アドレス方式の計算機のためのものである。2 アドレス，3 アドレスおよび 4 アドレスの機械も作られてきているが，基本的な考え方を研究した上で，これらに触れることにしよう。考え方を明確にするため，例として，1 アドレスの機械をとりあげてみよう。まず計算機を使って問題を解くには，プログラムと呼ばれる計算機の一連の動作を書き上げる必要があることを思い起こそう。プログラムの基本的な形は 2 章で述べたような機械語によるステートメント（文章）からできている。最初は穿孔カードやテープにこのステートメントが入っているが，実行する前には，これを計算機の記憶装置に入れる。プログラムの各ステップは，最初から 1 つ 1 つ順番に実行される。この際，分岐命令 (branch instruction) がきて，次からのステップがいくつか省略されたり，反復したりする可能性もある。記憶装置に入れられたプログラムの命令に対して，計算機はこれらの命令を処理するために

*〔訳注〕 論理回路のあとに設けられる増幅器で，選択された文字を目に見えるようにするために必要である。

さらに細かいいくつかのステップを実行しなければならない。まず命令を記憶装置から制御装置へ読み込む必要がある（FETCH）。制御装置の中では，この命令は分解されて，実行するための適当な制御信号（コマンドやタイミングなど）が作られる（EXECUTE）。基本的な FETCH-EXECUTE は，HALT または STOP 命令がくるまで間断なく繰り返される。図 9-11 に示すように，制御部の基本的な部分は，(a) 次に実行すべき命令（NEXT instruction）が入っている番地または場所を入れておく格納レジスタ（次命令レジスタ）と，(b) いま実行している命令（PRESENT instruction），すなわち，動作またはコマンドとオペランドの番地を含む命令を入れておく命令レジスタである。

FETCH の動作の間に，次命令レジスタに示された番地に入っている語（word）が記憶装置から命令レジスタに読み出される。そして次命令レジスタの内容は 1 つ進む。たとえば 152 番地にある命令がいま読み出されたとすると，次命令が入っている番地は 153 番地である。FETCH の動作が終ると，今度は EXECUTE の段階に入る。命令レジスタに入っている語の操作部分（operation part）がとり出され，制御回路およびタイミングの回路に，どのような信号を出せばよいか指示する。たとえば，操作の符号（OP コード）が 10 だとすると，解読回路は消去・加算（clear-add）のラインを能動状態にし，計算機のいろいろな装置でこの動作を実行させるためいくつかの刻時信号を出すことになる。

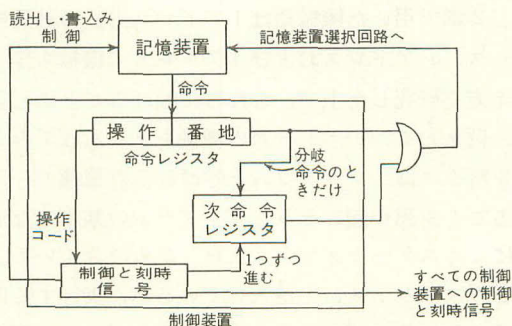



図 9-11　制御装置のブロック図

次に，命令のアドレス部に示された番地の内容を読み，演算装置のアキュムレータを 0 にした後，この内容をアキュムレータに加える。EXECUTE サイクルが終ると，次の FETCH サイクルが始まる。この例では，153 番地の内容が命令レジスタに入り，次命令 レジスタ の内容は 154 になる。

この手順は分岐命令のときだけ変更される。命令が無条件分岐（unconditional-

al branch, コード 44) だとすると，最初 FETCH の動作は前と同じで，次命令レジスタは1進む。たとえば，この命令が 362 番地から読み出されたとすると，次命令レジスタは 363 になる。しかし，EXECUTE のときに，無条件分岐であることがわかると，この命令の番地部がとり出され，これが次命令レジスタにダンプまたはシフトされる。もしこの番地が 325 だったら，いま次命令レジスタに入っている 362 は 325 にかわる。これで EXECUTE の段階が終り，機械は FETCH に入り，今度は 325 番地から1語を持ってくることになる。FETCHの動作に関する限りは，なんらかわったことは起こらず，計算機は続けて，325 番地以後の連続番地から命令を持ってくることになる。条件つき分岐命令（コード 45, 46, 47）であるとすると，アキュムレータを見て各条件（45 は負，46 は正，47 は零）が満足されていると，その命令の番地部を次命令レジスタに置く。もし満足されなければ，そのまま FETCH サイクルに入る。

制御装置は命令レジスタにある現在の演算（操作）コードに対する制御回路を選ぶために，解読ゲートやマトリックスを持っている。刻時信号は，いくつかの演算の型を作り出す計数器や解読ゲートからとり出される。これらは加減乗除などの動作を行なうよう結線されているものである。

2アドレス方式の命令コードを持つ計算機はいくらか違った動作をする。この命令コードは次のような3つの部分から成っている。すなわち，演算コード部，被演算番地部（operand address），次命令番地部（next instruction address）である。被演算番地部は，EXECUTE のとき，演算コードの被演算子として働き，次命令番地部は，FETCH のとき，次のプログラム命令を得るために記憶装置にその内容を提供する。各命令には次に行なうべき命令の番地が含まれているから，命令の番地を更新する必要はない。分岐令令では，2 つの番地部のうち，1 つが分岐の場合の次命令の番地の指定に用いられる。分岐条件が満足されたり，あるいは条件が存在しない場合には，次命令番地部が使われ，満足されないときは，被演算番地部が使われる。

3 アドレス命令では，2 つの番地部が算術演算の2個の被演算子を指定するのに使われる。たとえば，加算命令では，加えるべきAとBの番地を持っている。3 番目の番地部が次命令番地部となることもあるし，算術演算の結果をしまう場所を指定するのに使われることもある。4 アドレス命令となると，前のAとBの他に演算結果を格納すべき番地および次命令の番地を指定することができる。番地部が多くなれば，命令の構成は大きくなるが，プログラムのステ

ップ数は少なくなることは明らかである。磁気ドラム記憶装置を使うときには，読み出しの動作は周期的に行なわれ，平均して1回転の時間の半分待たなくてはならないから，命令の読出し回数が少ないと全体的に早い動作ができる。

HALT 命令がくると，計算機は，自動演算の状態から手動状態に切りかわる。こうなると，オペレータは機械を操作することができる。大型の計算機では，プログラム終了時に毎回機械を止めてしまうのは不経済であって，一般に1つのプログラムが終ると，直ちに次の新しいプログラムを要求するようになっている。記憶装置には，絶えず機械を稼動状態に保つため十分な異なるプログラムが入っているわけではないから，終了後分岐した先のいくつかの命令では，新らしいカードまたはテープを読み込み，この新しいプログラムへの分岐を指示する。誤動作が起こり，そのことを機械が検知できるときは，このプログラムの計算をやめ，別のプログラムを読み込むようにすることもできる。このように，大型の計算機では，連続的に処理を行ない，広い範囲の問題の処理ができるようになっている。

要　約

制御装置の基本的な部分についてこの章で述べてきた。これらの装置は，解読ゲート，循環計数器，ダイオード マトリックス などから成っている。演算の順序を制御するために一定時間間隔のパルスを発生している。

ダイオード マトリックス は，符号化に用いられていて，多くの入力線に対し，そのどれか1本の線上の信号を2進数に変換するのに使用され，あるいはまた，逆に，ある2進数をそれぞれ1本の線上の1つの出力にかえるためにも使用される。

基本的な計算機の周期（machine cycle）は，FETCH-EXECUTE の繰返しである。FETCH のときには，次に実行すべき命令を呼び出し，EXECUTE のときには，その演算を行なう。複数個の番地部を持つ機械があるので，これをまとめておく。

1アドレス命令	演算コード＋被演算番地
2アドレス命令	演算コード＋被演算番地＋次命令番地
3アドレス命令	演算コード＋被演算番地＋被演算番地＋次命令番地（または結果番地）

4アドレス命令　　演算コード＋被演算番地＋被演算番地＋結果番地
　　　　　　　　　＋次命令番地

問　題

1. OR または AND ゲートだけを用いて，3段計数器のすべての組合せを解読する回路図 (interconnection diagram) を作りなさい。

2. NAND ゲートだけを用いて，2段計数器に対する解読ゲートの回路図を作りなさい。

3. MOD 3 計数器の 3 つの計数を解読するダイオード マトリックスを作りなさい。

4. ダイオード マトリックス で，BCD を 4 段計数器の 2 進数に符号化する回路を作りなさい。

5. 図 9-12 は計算機の簡単な制御部を示したものである。命令レジスタは 2 ビットで，番地レジスタは 3 ビットである。命令は 4 種可能である。

00：加算
01：減算
10：負のとき分岐
11：格納

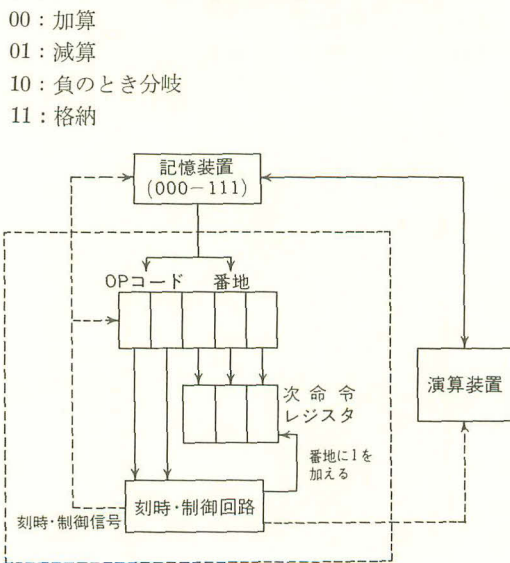



図 9-12　問題5の簡単な制御装置

3ビットでは，8個の記憶番地が決められる。この簡単な機械で，次のようなプログラムがあるとき，各ステップの FETCH, EXECUTE のサイクルで，何が起こるか詳しく述べなさい。ここでは，演算コードを増すことはしないで，プログラムを停止するためには実際の演算コードのかわりに HALT という語を使う。000 番地から実行開始し，そこではアキュムレータは 0 になっているものとする。

番地	命令: 演算コード	命令: 番地部
000	00	101
001	01	110
010	10	111
011	11	110
100	HALT	
101	01	111
110	11	111
111	HALT	

6. 問題5と同じ説明を次のプログラムについて行ないなさい。000 が開始の番地であり，そのときアキュムレータは 0 になっているものとする。

番地	演算コード	番地部
000	00	000
001	00	100
010	11	000
011	HALT	
100	00	001

1010＝10

電子計算機の算術演算

10-1　算術演算一般

電子計算機の算術演算装置には，加算，減算，乗算および除算を行なう論理回路が含まれている。計算機で処理しようとする情報は，一般に，まず記憶装置に入れられ，後に算術演算装置に送られる。そして算術演算の結果は記憶装置に戻される。算術演算の速度は速く（刻時単位は普通 1MHz* 程度），計算機の中にデータを入れる速度は遅い（1 秒当たり 2,3 サイクルから 10,000 サイクル位まで)ので,高低両速度の緩衝装置としての記憶装置の役割りは重要である。算術演算装置の高速性は“スクラッチパッドメモリ（scratch-pad-memory)”と呼ばれ，算術演算装置だけで用いる特別な高速記憶装置を必要とする。別に，低速大容量の記憶装置があって，全体の計算機能と，プログラムの格納のために使われる。

汎用電子計算機では，乗算と除算は，加算や減算を繰り返すことによって行なわれるので，算術演算には基本的には加算と減算の装置だけがあればよい。しかし，乗算と除算のためには，この他にたくさんの論理回路と，データを取り扱う回路が必要である。平方根や三角関数の値を求めるような複雑な計算は，表を格納（大きなデータ記憶場所が必要となる）しておいたり，または格納されているプログラムで，反復公式を使って計算したりすることにより行なわれる。それらの反復的な計算では 4 つの基本的な算術演算機能しか使わないようにしてあるので，論理ゲート，タイミングを制御することにより，また格納されたプログラムを使うことにより，すべての基本的な数学的演算は加算と減算

*〔訳注〕 1 MHz (Mega Hertz)。10^6 Hz。

により行なうことができる。

3章では，2進コードやその他のコードの数値について加算の操作を説明し，また1の補数をとって行なう減算についても述べた。どちらも，数の絶対値の計算だけで，符号は考えに入れていなかった。この章では，符号を考慮に入れた加減算について検討し，計算機で行なう乗算と除算の詳細について述べる。

第1に，レジスタ，加算器，論理ゲートを使って，算術演算の部分がどのように組み立てられるかについて考えよう。図 10-1 に典型的な算術演算装置の単純化したブロック図を示す。Aレジスタは記憶装置から情報を受けとる。そして，Bレジスタは算術演算の結果を記憶装置に入れる。加える2つの数を1

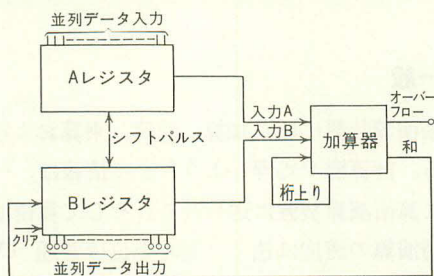



図 10-1　加算演算の単純化したブロック図

番低い桁からシフトして加算器を通すことによって，直列的な加算を行なう。加算器の出力として直列に得られる和は，Bレジスタにシフトして戻される。こうしてBレジスタには加算の結果が入る。減算を行なう場合には，加算器を減算器で置きかえればよい。しかし，データの流れは同じである。事実，和と差の表現は同じだから，借りまたは桁上りの項が異なるだけである。ブロック図では，入力データは並列に取り扱われ，加算は直列で行なわれている。より速く演算を行なうためには，記憶装置からデータをできるだけ速く出し入れすることが望ましいので，入出力の並列処理が必要とされる。もし速度が重要でなければ，データは直列に読んでもよい。その場合，Aレジスタは不必要になる。なぜなら，アキュムレータ（Bレジスタ）の中のデータを加算器にシフトするとともに入力データも記憶装置からシフトして加算器に加えることができるからである。速度が重要視されることが多く，たいていの大型の機械では，加算部分でも並列演算が行なわれる。最大の問題点は費用である。並列に加算を行なうには，大量の論理ゲートと加算器を必要とし，かなり高価なものになる。

オーバフロー信号は符号の決定に使われるが，その問題についてはあとで述

べる。図の桁上り信号は前のビットの加算による遅れた桁上り信号である。データは加算器に3つのステップで入れられる。消去信号により，Bレジスタを零にリセットする。加える最初の数はAレジスタに持ってこられ，直列に加算器にシフトされてBに加えられ，そしてBに格納される。そのあとで，2番目の数をAに持ってくる。これで算術演算装置へデータが入ったわけである。以上が基本的な型であって，Aへの並列転送，次に，AとB両方へシフトパルスを与えるということから成っている。加算を行なうためには，この型が3サイクル必要である。最初に，並列シフトにより，数を一時的にAレジスタに入れる。そしてシフトパルスでその数をBレジスタにシフトする（加算器を通して)。2番目に，別の数がAに入れられ，シフトパルスにより，2つの数が加えられ，結果はアキュムレータに入る。最後に，加算の結果は記憶装置に移される。最後のシフトは演算に何の影響もないので，ここでは無視する。以上の過程は非常にゆっくりしたものに思えるが，計算機の中では，すべての演算は一瞬のうちに行なわれる。ここでの例について，刻時周波数が1MHzと仮定すると，データ読込みに1マイクロ秒，10ビットの語をシフトするのに10マイクロ秒必要であり，加算の3サイクルで33マイクロ秒になる。したがって，1秒間に $1/(33\times10^{-6})=30,000$ 回の加算ができる。これだけの高速で演算できるというのが，計数型電子計算機の際立った特徴である。乗算や除算では，いくつものステップが必要となり，必然的に遅くなる。速度が重大な問題になるところでは，完全な並列演算の機械が使われる。逆に速度があまり重要でないところでは，直列演算を行なうことによって費用を安くすることができる。

乗算や除算を行なうためには，さらに2つのレジスタを使う。これらをCおよびDレジスタと呼ぼう。図10-2に必要ないくつかの論理の修正を示してある。詳細についてはあとで述べる。Bレジスタには乗数（multiplier)，Cには被乗数（multiplicand）が入っている。2つの10ビットの数の積は20ビットの数となるから，積はレジスタAとDに作られ，蓄えられる。Aには上位半分が入る。乗算を行なうのに，読出しとシフトについて，いままでと同じ基本的な型を使うことができる。まず，AとDレジスタを零にする。次に，乗数の入っているBレジスタの出力の桁の数字に応じて，最初は零に被乗数が加え込まれる。次のシフトで，この部分和は，乗数の次の桁から生じる積に加えられる。乗数が0のときは，Aに零を加える。乗数が1ならば，被乗数はAの部分和に加えられる。1桁ごとの桁移動が必要なことはあとで説明する。Cレジスタの中の被乗数は何度も使われ，乗算の間，そのまま保持しておかなければ

ならない。これはCの中の数を循環桁移動することによって行なわれる。循環桁移動は情報を保持したままにしたいときに使われる。いつ新らしい情報を入れるための桁移動をするのか，また前のデータを循環桁移動するのかを決定するための論理ゲートが必要である。

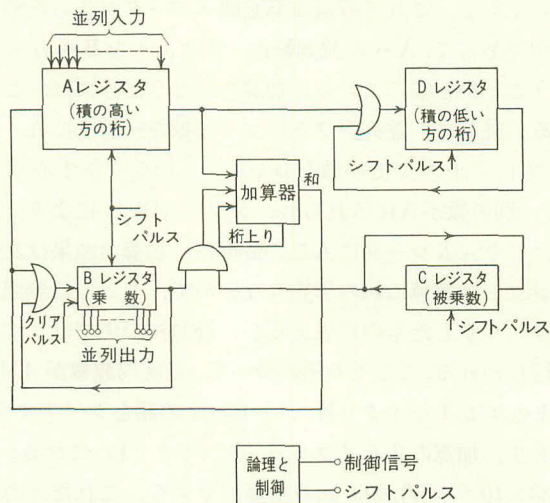



図 10-2　乗算における算術演算装置

10-2　算術演算装置，加算/減算

加 算　符号のついている2つの数を加えるとき，正しい答を得る操作が自動的に行なわれているように見えるが，ある種の決定が行なわれているのである。もし両方の数が同じ符号なら2つの数は加えられ，符号はそのままである。符号が同じでないとき，引き算が行なわれ，絶対値の大きい数の符号が答の符号に使われる。このことは，言うのは簡単で，また人間はこれを容易に行なっている。しかし，計算機でこのことを行なうためには，はっきり定まった演算の規則が必要である。どちらの数が大きいか，いつ大きい数を探すか，またどうやってそのことを算術演算装置に知らせるのだろうか。加算を行なう1つのすなおな方法は次のとおりである。最初に符号が同じかどうかを比較する。同じなら，数を加え，同じ符号を使う。もし同じでないなら，A−B と B−A を2つの別な減算器で行なう。オーバフローは被減数が減数より小さい場合に生じるから，そのときには，減数の符号を使い，他方の減算器の値を差として使う。

この方法は簡単な方法ではなく，よく使われる方法でもないので，混乱を避けるためにこれ以上述べない。この方法の変型として，減算器を使わないですむように負の数に1の補数を使う方法がある。手順は本質的には同じである。

符号のついた加算と減算を機械的に行なう簡単な方法を次に述べる。この方法のための規則は次のとおりである。0と1をそれぞれ正と負の符号を表わすのに用いる。正の数は，その数の絶対値の最上位に，正符号を表わす0をつけて表わす。負の数は，最上位の桁に1を入れ，その数の絶対値の1の補数を続けて書いて表わす。両方の数は，同数のビットを用いて表わす。このとき，和をとっても数の部分と符号の部分とが混ざらないように十分桁数をとるべきである。以上のことは，以下の例を見れば，はっきり理解できる。これらは手順のための規則であって，機械化を複雑にするものではない。手順全体は指定された書式を使い，数と符号をまとめて数とみなして，2つを加えるだけである。この手順を使って紙の上で計算するとき，正しい書式になっているかを確めなくてはいけない。数を書き表わしてしまえば，符号はないかのように，演算は機械的に行なわれる。答は先に決めた書式で得られる。

例 10-1 次の数を上に述べた書式で書きなさい。

(a) +7　(b) +12　(c) −3　(d) −9　(e) −21

解答：

(a) +7 は 0 111 すなわち 0111
　　↑ 正符号　111：2進数の7

(b) +12 は 0 1100

(c) −3 は 1 00
　　↑ 負符号　00：3の1の補数 ⟶ 3は $(11)_2$*
　　3の1の補数は $(00)_2$

(d) −9 は 1 0110 ⟶ 9は $(1001)_2$
　　1の補数は $(0110)_2$

(e) −21 は 1 01010 [21は $(10101)_2$]
　　↑ 負符号　01010：21の1の補数

例 10-2 次の2つの数を加えなさい。

---

*〔訳注〕数値の桁数を決めて補数表現をしないと誤解されるおそれがある。たとえば，4ビットとすると3は0011であり，その1の補数は1100である。したがって，−3は11100となる。

(a)　+6，+1　(b)　+9，+4　(c)　+2，+8　(d)　+8，+9

解答：

```
(a)   + 6           0 110
      + 1  ──→      0 001
      ────          ─────
      + 7           0 111     答：+7
(b)   + 9           0 1001
      + 4  ──→      0 0100
      ────          ──────
      +13           0 1101    答：+13
(c)   + 2           0 0010    両方の数の桁と同じになるように +2 は 0010
      + 8  ──→      0 1000    としなければいけない。
      ────          ──────
      +10           0 1010    答：+10
(d)   +8は0  1000
      +9は0  1001
```

この答には5ビット必要なので，数は5ビット使って書かなくてはならない(5 桁目の最上位のビットは零である。この場合，これは問題にならないが，負の数に対して補数をとる場合重要になる)。

```
      + 8           0 01000
      + 9  ──→      0 01001
      ────          ───────
      +17           0 10001    答：+17
```

問題を解く際，桁数には十分注意しなくてはならないけれども，加算器では桁数は決まっていて，データを自動的に正しく処理するようになっている。たとえば，10 ビットの加算レジスタなら，1 ビットが符号，1 ビットが桁上り，そして8ビットが加える数のためのものになる。この機械を使うときは，加算できる最大の数は8桁であり，加算する数が小さいときは，高い桁のビットは0となっていることを知っていなくてはならない。

例　10-3　次の2つの数（大きい方の数が正）を加えなさい。

(a)　+6，−3　(b)　+8，−4　(c)　+9，−6　(d)　+8，−5

解答：

```
(a)   + 6         0 110
      − 3         1 100        (3 は 011)
      ────       ──────
      + 3        10 010
                  └──→1        (循環桁上り)
                 ──────
                  0 011
                  ↑ └┬┘
             正符号  3         答：+3
```

(b)　+ 8　　0 1000
　　− 4　　1 1011　　(4 は 0100)
　　+ 4　　10 0011
　　　　　　　→1　　(循環桁上り)
　　　　　　0 0100
　　　　　　↑
　　　　　正符号　4　　答：+4

(c)　+ 9　　0 1001
　　− 6　　1 1001　　(6 は 0110)
　　+ 3　　10 0010
　　　　　　　→1　　(循環桁上り)
　　　　　　0 0011
　　　　　　↑
　　　　　正符号　3　　答：+3

(d)　+ 8　　0 1000
　　− 5　　1 1010
　　+ 3　　10 0010
　　　　　　　→1　　(循環桁上り)
　　　　　　0 0011
　　　　　　↑
　　　　　正符号　3　　答：+3

例　10-4　絶対値の大きい方の数が負である次の2つの数を加えなさい。

(a)　+9, −12　　(b)　+6, −7　　(c)　+5, −18　　(d)　+2, −4

解答：

(a)　+ 9　　　　0 1001
　　−12　⟶　1 0011　　(12 は 1100)
　　− 3　　　　1 1100
　　　　　　　　↑
　　　　　　負符号　3 の 1 の補数　　答：−3

(b)　+ 6　　　　0 110
　　− 7　⟶　1 000
　　− 1　　　　1 110
　　　　　　　　↑
　　　　　　負符号　1 の 1 の補数　　答：−1

(c)　+15　　　　0 01111
　　−18　⟶　1 01101　　(18 は 10010)
　　− 3　　　　1 11100
　　　　　　　　↑
　　　　　　負符号　3 の 1 の補数　　答：−3

```
(d)   + 2          0 010
      − 4  ──→     1 011          (4 は 100)
      ─────       ───────
      − 2          1 101
                   ↑ ︸
              負符号  2の1の補数     答：−2
```

例　10-5　次の2つの負の数を加えなさい。

(a)　−3, −4　　(b)　−9, −6　　(c)　−12, −10　　(d)　−8, −14

(e)　−13, −16

解答：

```
(a)   − 3         1 100
      − 4  ──→    1 011
      ─────      ───────
      − 7        10 111
                 └──→1          (循環桁上り)
                 ───────
                  1 000
                  ↑ ︸
             負符号  7の1の補数     答：−7

(b)   − 9         1 0110
      − 6  ──→    1 1001
      ─────      ────────
      −15        10 1111
                 └───→1         (循環桁上り)
                 ────────
                  1 0000
                  ↑ ︸
             負符号  15の1の補数    答：−15

(c)   −12         1 10011        (12 は 01100)
      −10  ──→    1 10101        (10 は 01010)
      ─────      ─────────
      −22        11 01000
                 └────→1         (循環桁上り)
                 ─────────
                  1 01001
                  ↑ ︸
             負符号  22の1の補数    答：−22
```

答に5ビット必要なので，12 と 10 は5桁の数で書いてある。問題が演算に用いる数と答とに同じ有効桁数を使って解ける場合，和のための余分の桁数は必要ない*。

*〔訳注〕余分の桁数は必要でないが，例によって説明する場合，十分の桁数を考えて，1語の桁数を規定することが望ましい。189ページに示したような誤解をなくすることができる。

```
(d)   − 8          1 10111
      −14  ───→    1 10001
     ─────        ─────────
      −22         11 01000
                   └────→1        （循環桁上り）
                  ─────────
                   1 01001
                   ↑  ︸
              負符号  22の1の補数    答：−22

(e)   −13          1 10010
      −16  ───→    1 01111
     ─────        ─────────
      −29         11 00001
                   └────→1        （循環桁上り）
                  ─────────
                   1 00010
                   ↑  ︸
              負符号  29の1の補数    答：−29
```

上の例でわかるように，循環桁上りの規則を使えば，符号の桁も他の桁の数字と同様に加えることができる。

問題 10-1　次の数を2進の形で書きなさい（例 10-1 参照）。

1．+3　2．−11　3．−17　4．+26　5．−19　6．+17

問題 10-2　次の2つの数を例 10-2〜5 の方法で加えなさい。

1．+6, +12　2．+8, −6　3．+6, −8　4．+13, −13　5．+9, −12
6．−8, −12　7．−2, +4　8．−16, +13　9．−10, −12　10．−21, +25

減 算　減算を上に述べた方法を使って行なうことができるが，少し複雑になる。詳しく説明するより，4つの型の例を示そう。新らしい技法は必要でなく，前の方法の完成ということになる。減算は1の補数を加えることによって行なわれる。

例　10-6　次の計算をしなさい。

(a)　+6 引く+4　　　(b)　+4 引く+6

解答：

```
(a)        +6             0 110             0110
       (−) +4  ───→  (−) 0 100  ───→  (+) 1011
      ─────────      ───────────      ──────────
           +2                               10001
                                                1
                                      ──────────
                                             0010    答：+2

(b)        +4             0 100             0100
       (−) +6  ───→  (−) 0 110  ───→  (+) 1001
      ─────────      ───────────      ──────────
           −2                                1101    答：−2
```

例　10–7　次の計算をしなさい。

(a)　+6 引く −2　　　(b)　+2 引く −6

解答：

(a)
$$\begin{array}{r} +6 \\ (-)\ -2 \\ \hline +8 \end{array} \longrightarrow \begin{array}{r} 0\ 0110 \\ (-)\ 1\ 1101 \\ \hline \\ \end{array} \longrightarrow \begin{array}{r} 00110 \\ (+)\ 00010 \\ \hline 01000 \end{array}$$
答：+8

(b)
$$\begin{array}{r} +2 \\ (-)\ -6 \\ \hline +8 \end{array} \longrightarrow \begin{array}{r} 0\ 0010 \\ (-)\ 1\ 1001 \\ \hline \\ \end{array} \longrightarrow \begin{array}{r} 00010 \\ (+)\ 00110 \\ \hline 01000 \end{array}$$
答：+8

例　10–8　次の計算をしなさい。

(a)　−4 引く −6　　　(b)　−3 引く −2

解答：

(a)
$$\begin{array}{r} -4 \\ (-)\ -6 \\ \hline +2 \end{array} \longrightarrow \begin{array}{r} 1\ 1011 \\ (-)\ 1\ 1001 \\ \hline \\ \end{array} \longrightarrow \begin{array}{r} 11011 \\ (+)\ 00110 \\ \hline 100001 \\ \hookrightarrow 1 \\ \hline 00010 \end{array}$$
答：+2

(b)
$$\begin{array}{r} -3 \\ (-)\ -2 \\ \hline -1 \end{array} \longrightarrow \begin{array}{r} 1\ 100 \\ (-)\ 1\ 101 \\ \hline \\ \end{array} \longrightarrow \begin{array}{r} 1100 \\ (+)\ 0010 \\ \hline 1110 \end{array}$$
答：−1

例　10–9　次の計算をしなさい。

(a)　−6 引く +2　　　(b)　−2 引く +6

解答：

(a)
$$\begin{array}{r} -6 \\ (-)\ +2 \\ \hline -8 \end{array} \longrightarrow \begin{array}{r} 1\ 1001 \\ (-)\ 0\ 0010 \\ \hline \\ \end{array} \longrightarrow \begin{array}{r} 11001 \\ (+)\ 11101 \\ \hline 110110 \\ \hookrightarrow 1 \\ \hline 10111 \end{array}$$
答：−8

(b)
$$\begin{array}{r} -2 \\ (-)\ +6 \\ \hline -8 \end{array} \longrightarrow \begin{array}{r} 1\ 1101 \\ (-)\ 0\ 0110 \\ \hline \\ \end{array} \longrightarrow \begin{array}{r} 11101 \\ (+)\ 11001 \\ \hline 110110 \\ \hookrightarrow 1 \\ \hline 10111 \end{array}$$
答：−8

問題　10–3　例題 10–6 から 10–9 に示した方法を使って次の計算をしなさい。

1.　+7 引く +3　　2.　−8 引く +4　　3.　−5 引く −6　　4.　+9 引く +12

5. −4 引く −2　　6. −6 引く +9　　7. +12 引く −8　　8. +15 引く −17

10-3 乗　算

電子計算機は，その速い演算速度で加算を繰り返すことによって乗算を行なうことができる。10 進数ではこの方法はきわめてわずらわしいが，2 進数ではそうではない。乗算の手順は，たとえば74×86なら，74 を 6 回加え，1 桁ずらして，74 を 8 回加えればよい。2 進数では数字が 0 か 1 だけなので，各桁での加算の最大回数は 1 で，10 ビットの数では，10 回加算のステップがあればよい。1 ステップ 1 マイクロ秒とすると，全部で 10 マイクロ秒すなわち 1 秒間に 100,000 回の乗算ができる。タイミングとかデータ桁移動などの問題があるため，実際にはもう少し遅くなるが，それでもまだかなり速い。そして作るのが容易なので，この方法は汎用電子計算機での使用に適している。人間が計算する方法と機械が計算する方法とを区別するために，前者を“鉛筆乗算”，後者を“機械乗算”と呼ぶことにする。各方法で同じ数を使って演算を行ない，両者の差異と，機械がどのようにこの演算を行なうのかを例示しよう。

例 10-10　鉛筆乗算で 2 進数 1011 に 1001 を掛けなさい。

解答：

1011	被乗数
× 1001	乗数
1011	
0000	部分積
0000	
1011	
1100011	積

例 10-11　例 10-10 と同じ計算を機械乗算でしなさい。

解答：

1.	1 0 1 1		
2.	1 0 0 [1]	←	乗数の桁の位置。
3.	1 0 1 1		加える（乗数の桁が 1 だから）。
4.	1 0 1	1	部分和と乗数を右に桁移動する。
5.	0 0 0 0	↓	(4) と (5) を加える。
6.	0 1 0 1	1	和はレジスタ A と D に入る。
7.	0 1 0	1 1	右に桁移動。
8.	0 0 0 0	↓ ↓	(7) と (8) を加える。

	A	D	
9.	0 0 1 0	1 1	
10.	0 0 0 1	0 1 1	右に桁移動。
11.	1 0 1 1	↓ ↓ ↓	(10) と (11) を加える。
	1 1 0 0	0 1 1	積はAとDにできる。
	Aに入る 高位の部分	Dに入る 低低の部分	答：1100011

上にあげた例でわかるように，機械による方法は，加算と右へ桁移動の繰り返しである。図 10-2 で言うと，B レジスタに乗数，C レジスタには被乗数が入る。最初の加算で被乗数はAの中の値（最初は零）に加えられる。そして，その和は再びAに入れられる。そこで，すべてのレジスタを右に桁移動する。A，B および D レジスタは 1 桁右へ桁移動するが，C は循環桁移動して被乗数がそのまま残るようにする。AとDを桁移動することによって，データは 1 桁移動し，次の和は 1 桁高いところに加えられる。さらに，B レジスタを桁移動することにより，次の乗数ビット（この場合 0）が次の加算時に被乗数を加えるか，加えないかを決定するのに使われる。この手順を 1 つのレジスタまたは語のビットの数だけ繰り返す。ここでは 4 回繰り返されている*(次ページ脚注参照)。

例 10-12 機械乗算で，10101×10110 を計算して，鉛筆乗算で検算をしなさい。

解答：

A	D	
1 0 1 0 1		被乗数。
1 0 1 1 0		乗数。
0 0 0 0 0		零をAに入れる。
0 0 0 0	0	右に桁移動。
1 0 1 0 1	↓	A に被乗数を加える。
1 0 1 0 1	0	
1 0 1 0	1 0	右に桁移動。
1 0 1 0 1	↓ ↓	A に被乗数を加える。
1 1 1 1 1	1 0	
1 1 1 1	1 1 0	右に桁移動。
0 0 0 0 0	↓ ↓ ↓	零をAに加える。
0 1 1 1 1	1 1 0	
0 1 1 1	1 1 1 0	右に桁移動。
1 0 1 0 1	↓ ↓ ↓ ↓	Aに被乗数を加える。
1 1 1 0 0	1 1 1 0	積

答：11100 1110

検算：

```
        10101
        10110
     ---------
        00000
       10101
      10101
     00000
    10101
     ---------
    111001110        答：111001110
```

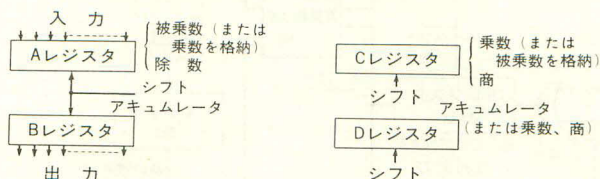
例 **10-13** 機械乗算で 101×11 を計算し，鉛筆乗算で検算しなさい。

解答：

```
 1 0 1 |
 0 1 1 |
-------+-----
 1 0 1 |          加算。
   1 0 | 1        右に桁移動。
 1 0 1 | ↓        加算。
-------+-----
 1 1 1 | 1
   1 1 | 1 1      右に桁移動。
 0 0 0 | ↓ ↓      加算。
-------+-----
 0 1 1 | 1 1                答：1111
```

*〔訳注〕 図 10-2 のようにレジスタを設定すれば，加減算においてBレジスタをアキュムレータと考えたように，乗算においてもBレジスタに積（和）の上位半分を入れ，下位半分を，たとえばDレジスタに入れるようにすればよい。このとき，記憶装置からAレジスタに被乗数を持ってくるか，または乗数を持ってくるかは乗算命令の設計によって異なり，前者においては，Cレジスタにあらかじめ乗数を入れておくべきであるし，後者の場合は，被乗数をCレジスタに入れておくべきである。また，積のみを求める場合には，Cレジスタは必ずしも必要としない。この場合，乗数をDレジスタに入れ，被乗数は記憶装置からAレジスタに入れられる。

乗除算のためのレジスタの役割

検算：

```
   101
   011
  ----
   101
  101
  ----
  1111
```

問題 10-4　次の計算を機械乗算でやり，鉛筆乗算で検算しなさい。

1.	2.	3.	4.	5.
110	1011	11010	10001	11011
× 10	× 1011	× 10010	× 1001	× 10100

10-4 除　算

除算は乗算と同じように，減算と左への桁移動の繰り返しで行なうことができる。この演算のためには，3つのレジスタがあればよい。というのは，答(商)には被除数と同じ桁数があればよいからである。乗算では，2つの10ビットの数の積は20ビットの数となり，別のレジスタが必要であった。除算は乗算に使ったレジスタを別の使い方をすればできる。図10-3に除算のための単純化した論理とレジスタを示す。言うまでもないことだが，大型の計算機では別の要請によって論理ゲートやレジスタの使い方が違うことがあるが，操作は同じで，減算と左桁移動の繰り返しである。図10-3はデータが除算でどのように扱われるかについてその基礎を知るのに役立つ。Aレジスタは算術演算装置の入力レジスタである。除数はAに読み込まれて，Bに桁移動される（Bレジスタは，はじめにクリアされている）。そしてAレジスタに数値（被除数）が入

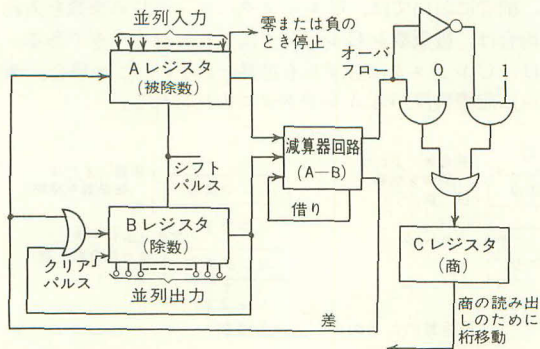



図 10-3　除算の単純化したブロック図

ると，除算はすぐに始められる*。A−B（被除数−除数）の減算器は差を計算し，オーバフロー出力を出す。もしオーバフローが0なら，除数は被除数より小さく，割ることができる。すなわち，被除数から1回だけ引くことができる。オーバフローのビットが1なら，減算の結果は負になる。すなわち，除数が大きくて，被除数から引くことはできない。Cレジスタには，各減算ごとに数字が入り，被除数の残りが零または負になったとき除算は終了する。

例　10-14　電子計算機の方法で，328を8で割り，鉛筆除算で検算しなさい。

解答：

1.		1010	01000	被除数（A）
2.		1000		除数（B）
3.		0010	01000	（1）引く（2）
4.		00100	1000	左に桁移動（1がCに入る。オーバフローなし）
5.		1000		（4）引く（5）
6.	オーバフロー ——→	11100	1000	オーバフローが起こり，Cに0が入る。
7.		00100	1000	（4）を再生する。
8.		001001	000	左に桁移動
9.		1000		（8）引く（9）
10.		000001	000	オーバフローなし（Cに1を入れる）。
11.		0000010	00	左に桁移動
12.		1000		（11）引く（12）
13.	オーバフロー ——→	11010	00	オーバフロー(Cに0を入れる)。
14.		00000100	0	（11）を再生して，左桁移動。
15.		1000		（14）引く（15）。
16.	オーバフロー ——→	11100	0	オーバフロー(Cに0を入れる)。
17.		000001000		（14）を再生して，左に桁移動。
18.		1000		（17）引く（18）。
19.		000000000		オーバフローなし（Cに1を入れる）。

*〔訳注〕除算の場合も乗算と同じように，BおよびDレジスタをテキュムレータと考えると，被除数をBレジスタ（およびDレジスタ）に入れて，除数をAレジスタに入れ，商をCレジスタ（あるいは，Cレジスタがない場合Dレジスタ）に入れればよい。

被除数の差はすべて零になった。ゆえに割り切れた。答は 101001。

検算：

```
     41
  8)328              答：41

        101001
1000)101001000
     1000
     ------
     001001
       1000
     ------
      0001000
         1000
     -------
         0000        答：101001
```

理論は簡単であるが，紙に書くと長くなる。手順をはっきりさせるため，次の問題をやってみなさい。

問題 10-5　次の数について機械除算をして，鉛筆除算で検算しなさい。

1．$101\overline{)11001}$　　2．$17\overline{)68}$　　3．$111\overline{)101010}$　　4．$9\overline{)81}$

要　約

算術演算装置は基本的演算すなわち加算，減算，乗算，除算を行なうための回路からできている。これらの演算を行なうためのいくつかの回路を示してあるが，実際には他にも多くの方法がある。

加算（減算）では，2 つの数は加算器（減算器）を通して移動され，結果の和（差）は出力レジスタに入れられる。出力は記憶装置に格納するために並列に読み出される。

乗算（除算）では，2 つの数は加算（減算）と右桁移動（左桁移動）により処理される。

0 と 1 を使って符号を表わし，それを最高位の桁に入れ，符号ビットも数として扱うことによって符号を持った 2 つの数の加算または減算を容易に行なうことができる。

問 題

1. 次の数を 2 進の形（例 10-1 参照）で書きなさい。

(a)　+27　　　(b)　−36

2. 次の数を 2 進の形（例 10-1 参照）で書きなさい。

(a)　+72　　　(b)　−56

3. 例 10-2〜5 と同様にして，次の 2 つの数を加えなさい。

(a)　+22, +16　　　(b)　+25, −15

4. 次の数について加算をしなさい。

(a)　−19, −33　　　(b)　−18, +15

5. 例 10-6〜9 と同様にして，次の計算をしなさい。

(a)　+18 引く（+8）　　　(b)　+25 引く（−16）

6. 次の数について減算をしなさい。

(a)（−36）引く（+14）　　　(b)（−24）引く（−12）

7. 次の数について機械乗算をしなさい。

$$\begin{array}{r} 1001 \\ \times\ 11 \\ \hline \end{array}$$

8. 次の数について機械乗算をしなさい。

$$\begin{array}{r} 11010 \\ \times\ 101 \\ \hline \end{array}$$

9. 次の数について機械除算をして，鉛筆除算で検算しなさい。

$$1010\overline{)1100100}$$

10. 次の機械除算を行ない，鉛筆除算で検算しなさい。

$$10\overline{)1010}$$

1011＝11

計算機の記憶装置

11-1　記憶装置概説

計算機の記憶装置は計算機がいかに有効なものであるかを言う場合にたいへん重要な役割を演じている。記憶装置の速度が速ければ，問題を解く速度も速くなる。また記憶容量の大きさは取り扱えるデータ量に関係し，複雑な動作ができるかどうかということに関連してくる。記憶装置の値段は，計算機の普及に関係がある。記憶装置の3つの要素，すなわち，速度，容量，および値段は記憶装置の型によってかなり異なっていて，記憶装置を選ぶうえで問題になるものである。

計数型の記憶装置は，2つの安定状態のいずれかをいつまでもとり続ける素子（電源が切られない限り）の集まりであると定義できる。フリップ フロップはこの定義で言うと，記憶装置の仲間になる。ところが，小さな計算機組織にすらかなりの記憶容量が必要であり，フリップ フロップ回路は大きさや値段がかさむため，大容量の記憶素子が必要なところには用いられない。一時に小量の情報を保持するようなところ，たとえば中間記憶装置すなわち計算機の装置間のバッファなどに使われている。記憶装置は数百ないし数十万語（1語は一定数のビットからなる）を含み，1語は数ビットのものから，36ビット，48ビット，64ビットのものがある。もし断わりがなければ，この本では，1語に10ビットあるものとして話を進めていくことにする。普通よく使われる記憶装置は，磁気コアと磁気ドラムであるが，磁気テープ，紙テープ，穿孔カードなどもまたこの記憶装置の仲間である。しかし，前の2つ以外は，主として，計算機の中にデータを入れたり，データを計算機からとり出して比較的長くとっておく場合に使われることが多く，入力装置という範疇で扱うのが最も適している。前の2つとあとの記憶装置のはっきりした差異は，演算装置で記憶装置を

直接用いられるかどうかということである。直接用いられるものを計算機の記憶装置と呼ぶことにしよう。この内部の記憶装置に送り込むデータを補給したり，内部の記憶装置からデータをとり出したりするものは入出力装置と考えられる。これについては，12 章で詳しく述べる。計算機の内部記憶装置には，あまり一般的ではないが，他に，光学記憶素子，トンネル ダイオード，多孔コア（トランス フラクサ，multiaperture core または transfluxor)，ツイスタ(twistor)，超低温記憶素子，薄膜記憶素子，フェライト コア などいろいろある。このうちいくつかについて簡単に触れることにしよう。これらは根本的には速く小さいが，一般に高価で，現在（1965 年 9 月）まだ研究中，もしくは限られた範囲内での使用にとどまっている。将来いつかはこれが主要な地位を占めるかもしれないから，これを知っておくことは重要である。現在広く使われている磁気コアおよび(それにはいくらか劣るが)磁気ドラムにはもっと詳細な使用上の考察を払う必要がある。さらにこれら 2 つは違った構成を持っており，記憶装置の利用上の重要な要因である記憶装置の型の論議にも役立つであろう。

記憶の動作を述べるのに，多くの表現のしかたがある。**ビット直列**というのは，1 つ 1 つのビットが順次すなわち一度に 1 つずつとられることを意味し，**ワード直列**は，語が逐次すなわち一度に 1 語ずつとってこられることを意味する。**ビット並列**は，いくつかのビットが一度に読み出されることを言い，**ワード並列**は，いくつかの語が一度に読み出されることを言う。**呼出し** (access) というのは，語またはビットを記憶装置からとってくる動作をいう。呼出し時間（アクセス時間）は，記憶装置からデータを持ってくる時間であり，記憶装置の速度にとって重要な因子である。たとえば，ある本のあるページを見つける時間を考えてみよう。もし，一度に 1 ページずつめくっていくのだと 625 ページを見つけるのは 27 ページを見つけるより時間がかかる。このような方法を「逐次的方法」という。これと同じように，記憶装置のある場所をさがすのに，あらかじめ決まった順で進んでいくようなものが**逐次呼出し**記憶装置である。これに対し，索引ができていて，あるページがどこにあるか直接わかるようになっていれば，その場所に関係なく同じ時間で見つかることになる。このような記憶装置の動作を**ランダム呼出し**と呼ぶ。明らかに「ランダム呼出し」の方が「逐次呼出し」より速い。記憶装置の型を選ぶに際して，速度と値段の間にかね合いがあるから，ビットや語をどのように組み合わせるか，また逐次呼出しにするか，ランダム呼出しにするかは特定の問題に適するよう選ばれている。

11-2　磁気コア記憶装置

磁気コアがいまのところ最も一般的な記憶素子であるから，最初にこれから話を始めよう。この記憶装置とその性質を論じる前に，磁気コアの記憶特性から述べる。磁気コアは，普通ドーナツのような形状で，非常に小さいので裁縫用の指ぬきの中に何千と収まるほどである。これは特殊な性質の強磁性体でできている。そして，このコアには電流を通す線が数回巻いてある。これは磁束を作るためで，電流がその電線を通って一方向に流れると，フレミングの右手の法則に従って，磁束が生じる（図 11-1 参照)。電流を増加させていくと，

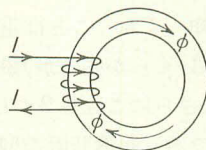



図 11-1　磁気コア内の磁束流

最初のうちは磁束は直線的に増加していくが，だんだん増加しなくなってくる。そしてついには，いくら電流を増しても磁束は増加しなくなる。この点で，コアは飽和（saturate）したといわれ，電流が増加しても，もはやこれ以上の磁束の増加は認められない。実際は，非常にわずかであるが増加しているのだが，計算機の目的に対しては，この増加は無視できるほど小さいのである。このようにしてコアは信号を“蓄えた”あるいは“記憶した”ことになるわけで，電流をとり去っても磁束は消えない。もちろん，2 進記憶装置として使われるには 2 つの状態がなくてはならない。

コアの巻線の電流の方向を逆にしてみよう。もし電流が増加すると，磁束は減少し，やがて 0 になり，さらに負になっていく。そして磁束は負の方向すなわち逆方向で飽和する。電流をとり去っても残留磁束が逆方向に残る。“負”というのはコアの両極がかわって，磁束が反対方向に通るということである。普通，磁束は N から S へ流れると考えられる。

さて磁気コアで重要なことは，コアの状態をどのように「読み出し」たらよいかということである。コアは磁化され，磁気コアにはどちらかの方向の磁束が通っている。すなわち，2 進法の ZERO によって示されるものと ONE によって示されるものとである。どちらの方向に磁化されているであろうか。磁束の大きさに変化がないと，磁束の方向を知ることは容易でない。方向を検知

するためには，磁束に変化を与えられるようにしなくてはならない。磁束の変化によってだけ，ある電圧を誘起することができる。しかし，磁束を変化させると，その磁気コアはもはや以前のように磁化していないことになる。格納された情報を読み出すために，コアの内容を「破壊」してしまう必要がある。これは**破壊的な読出し**と呼ばれている。同じ情報を何度も使えるようにするのには，読出しのあと，データを「再書込み」するという動作が必要になる。以上のような一般的な知識を心にとめ，もとへ戻って磁気コアがどのような働きをするか，記憶組織としてどのように構成されるかをもっと詳しく調べてみよう。

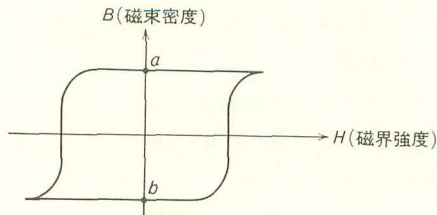



図 11-2 記憶用の磁気コアの *B-H* 曲線

計算機の記憶装置に使われているフェライトコアは図11-2にあるような「矩形」特性を持っている。点aとbは駆動信号をとり除いたあとに残る磁束密度である。a点を1，b点を0として扱っていく。コアを2進の1つの状態からもう1つの状態に「スイッチ」するためには，正しい極性と強さを持つ駆動信号が必要である。図11-3(a)に示してあるコアと駆動巻線について磁束と

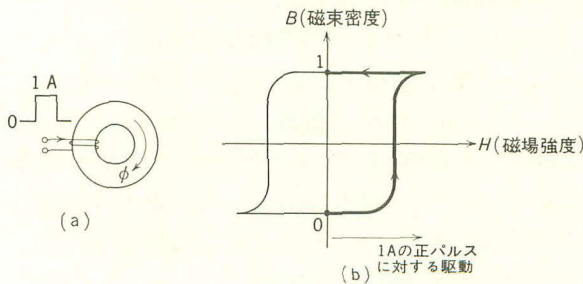



図 11-3 正の駆動（0 から 1）

電流の関係を考えてみよう。電線はコアのまわりに巻かれている。右手の法則により磁化する方向を知ることができる。コイルの巻き回数が決まっていると，電流は磁界の強さに直接比例する。図11-3(b)は，コアが入力の信号により磁化される様子を示したものである。駆動信号が0に戻ってからも，コアはONEの状態を持ち続ける。同じコアに逆方向にパルスを与えるとコアはZERO

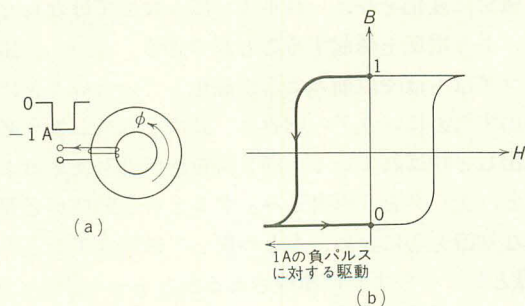



図 11-4 (1から0への) 負の駆動

の状態に戻る(図 11-4 参照)。

コアにパルス信号が入ると，その状態を切り替えるのにある一定の時間がかかる。その時間は駆動パルスの強さや変化速度によって異なる。速いコアでは，この時間は1マイクロ秒以下であり，遅いものでもほぼ10マイクロ秒である。このように磁気コアに対する呼出し時間は 1〜10マイクロ秒程度である。ここでは1マイクロ秒のものとしよう。磁気コアを記憶装置に使うとき，正しい極性を持つ電流信号によって ONE の状態にセットしたり，ZERO の状態にリセットしたりすることができる。コアは可能な限り小さく作られているので，巻線は一回だけで十分であり，その線は普通人間の髪の毛くらいの細さである。記憶装置をもうこれ以上小さくできないほど小さくするため，できるだけ少ない線でコアを働かせるのが望ましい。

さて，コアから読出しを行なうと，記憶が破壊されてしまい，コアは以前の状態を保持できない。読出しの信号によりコアの状態がかわると，読出し線によって検知され，コアには ONE が格納されていたとみなし，かわらなかったとき，コアに ZERO が格納されていたとみなす。ONE から ZERO へかわったときの磁束の変化により，読出し線には一定の電圧が誘起され，その磁気コアに，ONE が格納されていたことを知ることができる。もし，この誘起電圧がなかったら，コアには ZERO が格納されていたことになる。しかし，どちらの場合も読出しのあとには，コアの状態は ZERO になる。このような読出し方では状態がまったく破壊されてしまうので，「破壊的」であると言う。コアに格納されていた ONE は，1回だけ読出しが可能で，再び読もうとすると，今度は ZERO が読み出されてしまう。これに比べて，フリップフロップの記憶は読出しの信号がきても状態は変化しないようにできている。コアは破壊的であるため，一度読み出したら，いま読み出したものをもう一度コアに格

納する必要がある。このことは，コアの記憶装置の根本的な動作に，読み出し・書込みのサイクル (read-write cycle) が必要であるということを意味する。ONE が読み出されたことがわかると，ONE を再び書き込む（格納する)。一方，ZERO が読み出されたときは，コアは何もしなくてもよい。コアの内容を消す（リセットする）には，読出しの動作だけでよい。

コア記憶装置を構成する際，第一に考慮すべきことは，データがどこに格納されているかを追跡する方法である。ある特定のコアを見つけるのに，1 つ1つのコアに名前をつけるのが1つの方法である。これはビット単位のコア記憶装置である。これに対し，必要な選択量を少なくするために，コアは語（たとえば，1 語につき 10 ビット）にまとめられ，語を単位にして選択する。こうすると，読み書きの回数は 10 分の1になる。これが語配列のコア記憶装置である。ビット配列および語配列の両方における問題や解法について次に考えることにしよう。

コア記憶装置のデータを構成するにあたって主要な2つの問題点は，読み，書きおよび選択に必要な電線の量と，記憶装置の制御や操作をするのに必要な大量の選択回路とである。個々のコアについて読み，書きおよび選択の電線が必要である。1024 語の記憶装置では，3000 本以上の電線を必要とする（1024 は 2^{10} である。普通，語の数は2のベキ乗にすることが多い)。コアがいくら小さくても，3000 本もの電線があると，かなり大きな装置になってしまうので，x-y 座標による選択方式が用いられる。ある特定のコアを選ぶとき，x と y を指定して実際のコアをさすようにしたもので，図 11-5 に 100 個のコアがあったときの配列を示してある。12 番のコアを選ぶには，x_1 と y_2 を指定するだけでよい。どちらか1つでは不十分である。他のコアは2つの選択信号を受けないから，コア 12 だけが選ばれるのである。もし100個のコアに1本ずつ

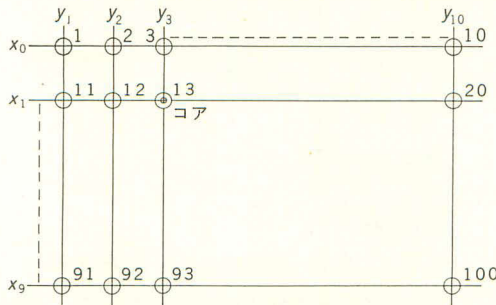



図 11-5 x-y 座標選択方式を用いた一般的なコア配列

選択線を通すのだと，100 本の電線が必要になる。*x*-*y* 座標方式を用いるとこれが 20 本あればすむ。これは 4/5，すなわち 80% もの節約になる。

1024 個のコアの場合には，1024 本の電線に比べて 64 本の電線ですみ，実に 15/16，すなわち，94% もの節約になる。コア記憶装置が大きくなればなるほど，*x*-*y* 座標選択方式は有利になってくるというわけである。このため，この方法はすべての大型計算機に使われてきている。*x*-*y* 選択方式を用いると，ハードウェア，すなわち回路の面でもっと別の節約ができる。100 個のコアに，100 個の駆動ゲートが必要なのだが，*x*-*y* 選択方式だとわずか 20 個のゲートがあればよい。

さて，コア選択の実際問題に目を移すと，この *x*-*y* 方式においては，1 つのコアを選ぶのに 2 つの信号が同時に必要である。コアは，ある一定の大きさのパルスを入れるまで磁化の状態をかえないという性質をもっている。選択パルスの強さを適当に定めて，2 つ合わさると十分な大きさになるが，片方だけでは不十分なようにしておく。図 11-6 はコアの *B*-*H* 曲線にこの様子を書き込んだものである。このパルスは電流パルスであって，一般に**電流一致方式** (coincident-current selection) として知られている。各選択線には，半分の選択電流しか流れないようにしてあり，1 つのコアで 2 つの半選択電流が一致すると，そのコアの状態を切り替えるわけである（*x* と *y* の両方向から信号がこない限り，コアはその状態をかえない）。一方，コアの状態を読み出すには，同じビット位置にあるすべてのコアに通した 1 本の電線によって簡単に調べられる。

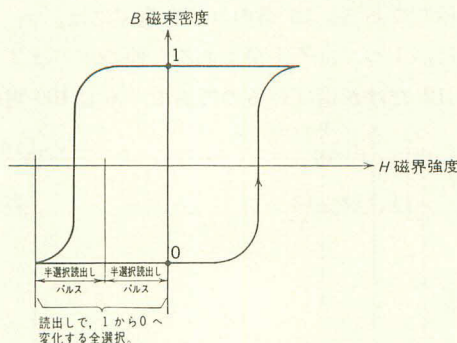



図 11-6　電流一致方式による *B*-*H* 曲線

選択回路によってどのコアが読まれているかがわかる。選択信号がきて磁束が変化すると ONE が入っていたことになり，電圧パルスが生じる。また磁束

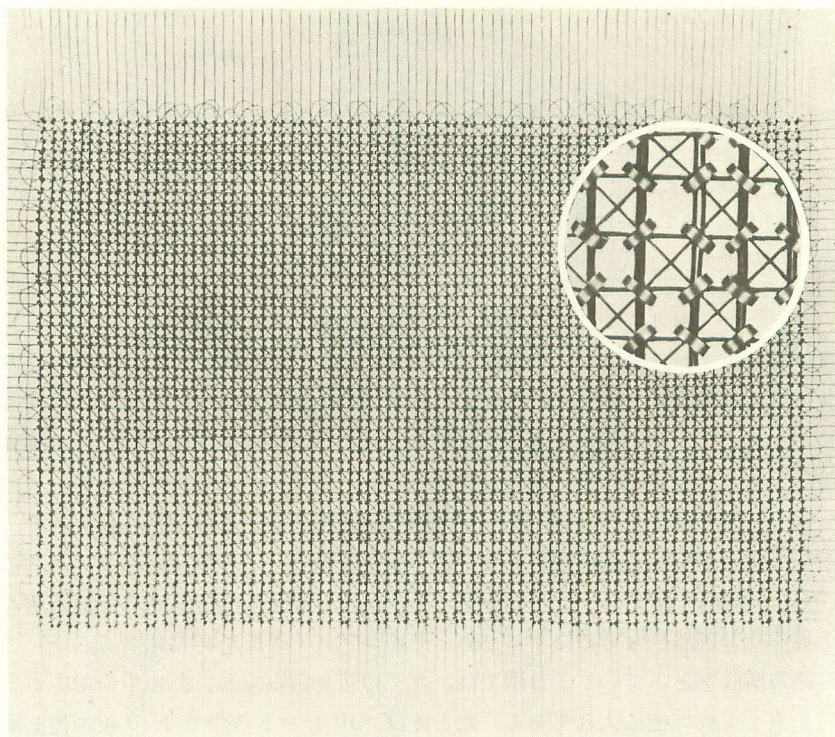



図 11-7 磁気コアの板

が変化しなければ ZERO であり，電圧パルスは生じない。読出しのときは，同じ信号を再び格納する必要があるが，この2本の読出し線を用いることができる。ONE を格納するには，この選択の線（2 本とも）に反対の極性の電流を与えてやればよい。反対方向の電流によって反対（ZERO に対して）方向の磁束を生じ，そのコアを ONE の状態にする。図 11-7 は磁気コアの板と巻線の詳細を示した拡大写真である。

1024 個の実際のコア配列は，図 11-8 に示されている。これは 32 行，32 列の配列になっていて，それぞれのコア線を操作する駆動回路がある。

選択が容易にできるように，32 個の列の1つを示す 5 ビットの数を蓄わえるバッファ レジスタ と，選ばれた 1 個の駆動回路をとり出すための**解読マトリックス**が備わっている。ある特別のコアを指定するには，行選択に 5 ビット，列選択に 5 ビット合計 10 ビット必要である。解読マトリックスは，特定のコアに対して1つの行と1つの列を選び，読出し信号をこの線に送る。この結果，

図 11-8　実際の電流一致方式による磁気コアの記憶回路

選ばれたコアだけが2つの半選択信号を受け，切り替わることになる（ONE の状態にある場合）。検出のためには，1 本の長い電線がすべてのコアに共通に使われていて，その線は半選択信号により他の 63 個のコアから現われる雑音電圧を拾うのを防ぐように巻かれている。

図 11-8 では，コア（10, 10）が太線で示してあるように選択されていて，電圧パルスが読出し線に現われたことを示してある。この電圧パルスはそこに ONE が格納されていたことを示す。このあと，書込みの動作で，記憶装置の全操作を非破壊的なものにするため，そのコアに ONE を再格納する。

語の構成をとっている計算機では，コアの板を積み重ね，語の中の対応するビットを 1 枚の板に入れてある。たとえば，10 ビットの語では 10 枚の板を要する。それぞれの板に 1024 個のコアがあれば，1024 語が格納される。1024 ビットのものが 10 枚あれば，そこに 10,240 個のコアが使われている。磁気コアや配線や回路は高価なので，この記憶装置は大がかりな投資にほかならない。おそらく，このコア記憶装置は，一般に用いられているものの中で最も高価なものであろう。蛇足的であるが，これの実際の大きさは 1024 ビットの板で $8\frac{1}{2}\times 11$（平方インチ）の本（ほぼ A 4 版）の $\frac{1}{4}$ 以下の大きさだし，最近は，4096ビットでこれ以下というものまでできている。厚さに関しては，配線のようなものも含めて，いまあげた本の約 10 分の 1 である。

読出しのあと，短時間のうちに書込みを行なわなければならないが，そのときには，出力信号は過ぎてしまっている。普通は，読み出された信号を フリップ フロップ レジスタ に格納して，それを使って書込み動作を行なうという方法がとられている。これはまた，バッファレジスタの役目をも果たし，計算機は前もって要求していた情報をそこから得ることができるようになっている。というのは，バッファ レジスタ があると，計算機が何か他の動作をしながら

記憶装置から情報を読むことができる。すなわち，読出しの操作を要求して，記憶データが完全にとり出される前に別の動作をしたりすることができるのである。読出しには再書込みがつきものであるから，読出しのときにはいつも，自動的に両方の動作をするよう制御回路を結線してある。このようにして，読出しの動作が非破壊的なものになる。しかし，書込み動作自身では，新しいデータが記憶装置に入り，古いデータは失なわれる。

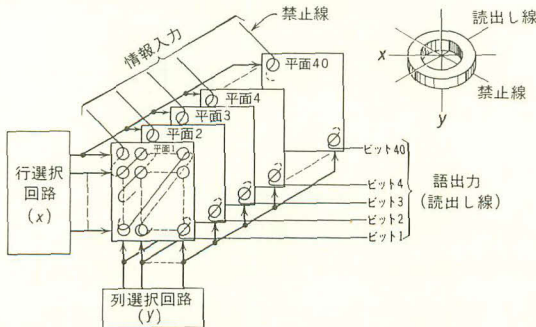



図 11-9 語構成のコア記憶装置

図 11-9 は簡単な語構成のコア記憶装置である。それぞれの板は，1024 ビットの板で図 11-7 のものと同種のものである。一致電流は，板の上の 1 つのビットを選択する。図 11-9 に示されているような 40 枚の板について，それぞれの板の 1 つのコアに読出しの信号が与えられる。各コアに 1 ビット蓄えられているので，これは 40 ビット語の装置である。読出しにより，40 ビットの情報が一度に読み出され，40 本の出力線に現われる（1 枚の板に 1 本）。

書込み命令には，1024 語のうちどれが選択されるかを規定した番地が書かれており，書き込まれる情報は，z 軸と呼ばれる第 3 の電線に送り込まれる。各板に 1 本ずつあるこの電線は，論理的には検出の電線と同じである。一般には，実際上の理由から別の電線を用いている。個々のコアでは，x, y, z の選択パルスにより生じる磁束は代数的に加えられる。これらのパルスは正または負の極性を持つものであり，時計まわりまたは反時計まわりの磁束が生じる。2 つの正の半選択信号があると全体として時計まわりの磁束となり，これは信号が終ったあともそのまま残る（図 11-10a 参照）。もし，2 つの信号が負の極性を持つものならば（図 11-10b），反時計まわりの磁束が生じて信号の通過後も残る。

第 3 の選択線 “z” はまた**禁止線** (inhibit wire) とも呼ばれ，負極性の半分の電流パルスを流すか，まったく流さないかいずれかである。電流が 0 ならば，

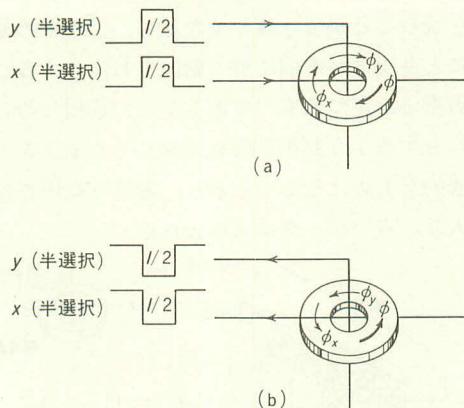



図 11-10　半選択パルス

もちろん何もしない。もし負の半分の選択電流が流れるなら，x または y の線に流れる正極性の半選択電流を妨げることになる。これがどのように用いられるかを見るために，記憶装置の１つのコアを読み出したり，書き込んだりする動作全体を考えてみよう。時計まわりの磁束があるとき ONE が格納されていると定義し，反時計まわりを ZERO が格納されていると定めておこう。読み出し信号はそこに ONE が入っていれば磁束を逆方向にし， ZERO が入っていればそのままにしておくよう，負の極性を持つものでなければならない。負のパルスが両方に入ると反時計まわりの磁束が生じ，もし ONE があったときには大きな磁束変化となり，もし ZERO があったときにはほとんど磁束の変化は起こらない（図 11-11）。読出しのときには，禁止線にパルスを与えない。こ

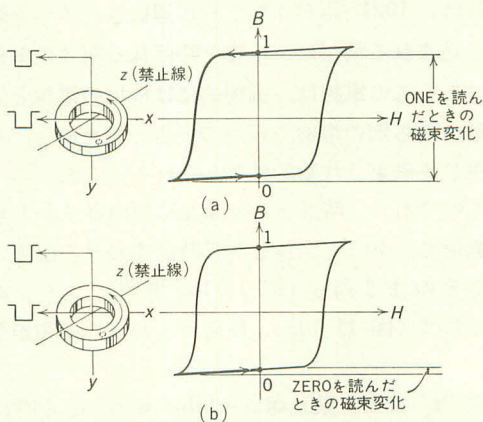



図 11-11　1つのコアの読出し

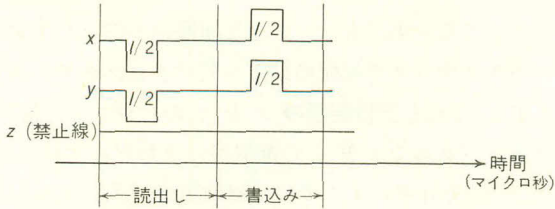



図 11-12 ONE を記憶するための読み・書きサイクル

の線は，要求されたビットの格納のため，書込みのときにだけ働くのである。

書込みの動作について考える前に，もう1つ明らかにしておかなければならないことがある。記憶装置の動作は，読み・書き のサイクルにまとめられている。このサイクルはデータの非破壊的読出しのため必要なのである。読出しのあとに書込みがあるのだから，書込みをしようと思ったら，その前に読出しをしなくてはならない。読出しは，読まれた語のすべてのコアを ZERO にしてしまうから，書き込む前にコアはすべて ZERO の状態になったことになる。これを考えれば，禁止線が必要なビットを書き込むのにどう使われるか見当がつくであろう。禁止線に信号がきていないと，x と y の選択線はコアに時計方向の磁束を作り，ONE を格納する。図 11-12 は，ONE が格納されるときの読み・書きサイクル を示す図である。

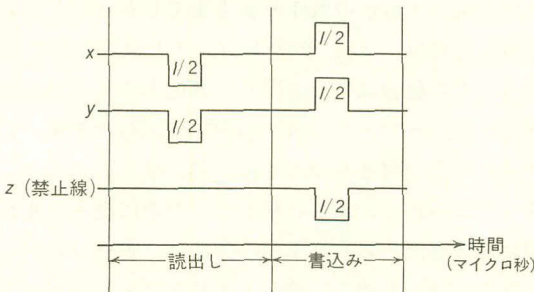



図 11-13 ZERO を記憶するための読み・書きサイクル

禁止線に負方向のパルス（図 11-13 参照）がきていると，1 つの半選択信号の磁束に反対することになり，書込み時の変化を禁止することによって，先行する読出し動作のために ZERO になっていたコアをそのままにしておく。メモリサイクルは，この 読み・書きサイクル から成っていて，現在の計算機ではたいていマイクロ秒の程度で行なわれている。

語構成のコア記憶装置の詳細について考察してきたので，これから完全な記憶装置の働きについてまとめて扱うことにする（図 11-9 参照）。記憶装置を

別々に働くものとして考えず，もっと大きな組織の一部として考える。

プログラムのあるステップで記憶の動作が要求されたとき，制御装置はその語の番地をとり出し，それを記憶装置の x と y のレジスタに送り込む。x と y の2つの解読マトリックスで x と y の選択線のうちそれぞれ1本の線を選ぶ。したがって，x と y それぞれ1つずつの駆動回路が選択される。制御装置は記憶の動作が行なわれるべきことを示し，新しいデータを書き込むのか，それとも記憶されたデータを読み出すのかを判別する。読出しであったら読出し信号を全駆動ゲートに送る。それぞれの板にある選択された x と y の駆動ゲートだけが負の読出し信号を出す。そして各板の読出し線は，記憶されていたビットが何であったかを示す。このとき書込みの信号が現われるまで維持するため，情報をバッファレジスタに格納する。書込みのときには，やはり同じx と y の駆動ゲートが選ばれるが，今度は入力パルスは正のものである。しかし，禁止線も働いて，読み出されたビットに従って，この線にパルスが流れないか，または負のパルスが流れ，それぞれ ONE または ZERO の書込みができる。読み出されたデータはバッファレジスタに格納され，計算機がそのデータを必要とするときはいつでもそこからとり出すことができる。当然のことながら，読み出された語はなお記憶装置の中に記憶され続けているが，バッファの中のデータは次の読み・書きの動作のときまでしか存在しない。

書込みの動作のときには，x と y のレジスタとマトリックスが働き，記憶装置の語を決める。この場合にも，読出し・書込みのサイクルがあり，読出し時に語の中のデータはバッファレジスタの中に入れられるが，しかし使われることはない。サイクル中の書込みのときは，禁止線への入力は，計算機の他のレジスタから持ってこられて，新しい語がコアの中に読み込まれる。新しい語の情報を禁止線に与えるためのもう1つの方法は，書込みの前に，バッファレジスタにそれを置き，読出し線上に読み出されたデータがバッファレジスタに入いるのを阻止するという方法である。その後，書込み動作を普通のようにすれば，新しい語をコアに入れることができる。読出しのときと同じ操作ですむ。

コア記憶装置の特徴をまとめると，読み・書きのサイクルは約1マイクロ秒で，大きさは非常に小さいが，値段は高い。ランダム呼出しができて，語の選択は非常にすみやかに行なうことができる。一般的に使われているコアの配列方法は x-y 座標を使った電流一致選択方式である。

11-3 磁気ドラム記憶装置

ドラム記憶装置は，コア記憶装置とはまったく異なっている。コアは非常に速いが，ドラムは遅い。コアはランダム呼出しができるが，ドラムは循環的である。コアが独立なビットの記憶素子であるのに対し，ドラムは大きな記憶の面である。コアは高価なのに，それと同容量の情報を入れるドラムは安価である。コアは比較的小さいのに，ドラムはかなり大きい。ドラム記憶装置は円筒状で，その表面には磁性体が塗布してある。データは 図11-14のようにドラム表面の円周方向に列状に格納され，特殊なヘッドで読み・書きされる。情報は小さな磁化された場所に蓄えられる。ドラムは一定の速さで回転していて，1回転に一定の時間かかる。1 分あたり 12,000 回転するものならば，1 回転に必要な時間は次のようになる。

$$T=\frac{1}{\text{速度}}=\frac{1}{12,000}\text{分/回転}\times 60\text{秒/分}=\frac{1}{200}\text{秒/回転}$$

すなわち

$$T=5\text{ミリ秒/回転}$$

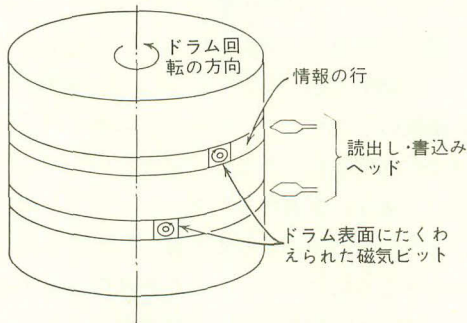



図 11-14　磁気ドラム記憶装置の概略

これはドラム記憶装置の現実的な速さであって，これをコア記憶装置の5マイクロ秒と比べてみると，1000 倍も遅い。これはかなり大きな違いである。データが必要なときにそのデータがヘッドのすぐそばにあるときにはすぐ得られるが，ちょうど通り過ぎたあとなら5ミリ秒かかる。平均して，1 サイクルの半分2.5ミリ秒である。これでもまだコアの呼出し時間とはかなり開きがある。だからドラム記憶装置を使う計算機は遅い機械ということになる。これは回路

が遅い刻時周波数で動作しているという意味ではない。ビットはドラムから1秒間に百万個の速さでとり出されるが，必要なデータをとり出そうとすると，平均数ミリ秒遅れてしまう。これにほとんどの時間がかかり，問題解決の時間は低下することになる（実サイクル時間が5ミリ秒で，1トラック（1列）に5000ビット入ると，5ミリ秒ごとに5000ビットだから，$5000/(5\times10^{-3})=1\times10^{6}$ すなわち毎秒100万ビット通過することになる)。

ドラムの回転速度は機械的な駆動機構によって制限を受け，それは記憶ドラムの大きさと重さに関係する。実際のサイクル時間の限界は，現在約0.5ミリ秒である。ドラムの表面にかなりなデータを入れようと思うと，それだけドラムの直径は大きくなければならない。いまのところ直径12インチのものが平均的なものとされている。ドラムは大容量の記憶装置によく使われるため，たくさんのトラックを入れなければならず，円筒の高さは約8インチである。1インチあたり20トラック，単位トラック長さ（インチ）あたり200ビット位（8ビット/mm）が典型的なものである。平均のものは，1インチあたり10万ビット位の容量である。10万個のコアを並べたたくさんの板の複雑さと比べてみるといいだろう。ドラムは1つのトラックにつき，読み・書き用のヘッドが1つだけついているので，数千ビットに1個の駆動回路あるいは読出し回路だけで処理できる。選択回路のゲートは数十個のトラックから1つのトラックを選択するだけでよい。操作が簡単なのでビットあたりの価格はコア組織と比べてかなり安くなり，ドラム組織の主な利点となっている。容量の大きいこと，安価であることがドラム組織の主な利点で，これこそ考慮すべき主眼点である。容量や価格が主な考慮すべき問題のときには，望ましいことではないが低速も容認しなければならない。速度が重要だと考えるならコアを用いればよい。システムによってはコアとドラムの両方をうまく使いわけているものもあり，大容量の記憶を必要とするところにはドラムを，高速動作を要するところには「一時的に記憶」しておくために小容量のコア記憶装置（スクラッチパッド記憶装置）と組み合わせて使用する。この2つが一緒になって働き，その結果計算機はいつもその最高速度で動作して，コアとドラムが互いに助け合うのである。

図11-15は読み・書きの機能の簡単な図である。読出しと書込みは別々に考えることができる。ONEまたはZEROを書くのには，書きたいトラックの書きたいビットの位置にきたとき，ヘッドから磁束を出して小部分を磁化すればよい。書込みヘッドに流す電流の極性を制御すれば，磁束の方向は決まる。時計方向をONE，その逆の方向をZEROとする。ドラム上の磁束の量

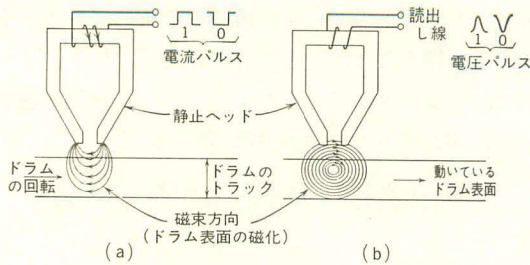



図 11-15 磁気ドラムの読み・書きの機能
(a) 書込みと (b) 読出し

は，電流パルスの強さ（その他に，コイルの巻数，ヘッドのコアの透磁率，表面とヘッドの間隔などの因子）で決定される。読出しのときの誘起電圧の量はドラムの表面に格納されているビットの磁界の強さ（回転速度，空隙，巻数，表面とヘッドの間隔などの因子）に従う。表面とヘッドの間の間隔はとても重要で5ミル (mil) (5/1000 インチ，すなわち 127 ミクロン) 以下が標準である。高速の機械では，**浮動ヘッド**といって，空気の圧力でヘッドと表面の間の間隔を保っているものもある。これだと，表面が熱で膨張するときにも，空気の圧力でヘッドと表面の間隔が一定に保たれているわけである。もし，ヘッドが1ミル離れて固定されているとき，過熱で1ミル膨張すると，ドラムとヘッドが触れ合って，ドラムのその部分が駄目になってしまう。浮動ヘッドは，この改良策であるが，機械的な複雑さもあり，価格も高くなる。

サイクルごとにヘッドによりトラックの上にデータが連続して格納されていく。データの番地づけが単純になるように，情報はある一定の出発点から読み出されるようになっている。番地はドラムに入れておかなくてもよい。トラック6の 27 番目の語の内容を知るには，出発点を検出する必要がある。外部回路が用意されていて，希望のトラックからきた語の数を数え，望む場所がきたときに，読込み開始の命令と終了の信号が正確なときに出される。

情報は一般にビット直列に並べられている。たとえば，1 列の続いた 10 ビットが所定の語を表わす。トラックは同時に読み出すことができるので，データは，語並列に処理できる。1 つのヘッドでも，トラックからトラックへそれを移動してすべての情報を読むことができるが，このようにすると，さらに動作速度をおとすことになるし，ヘッドの機械的精度を低下させるので，トラック間の分離が悪くなる。そのため，トラックごとに定まったヘッドをつけるのが普通である。

磁気ドラムの構成　ドラム上のデータの構成の仕方にはいろいろある。1 つのトラックに直列に語を作ることができる。たとえば，1 トラックに 4096 ビットあり，これが 32 トラックあったとしよう。ビット直列の機械（図 11-16a）では，1 トラックに 32 ビットの語が 128 語入る。

$$4096\frac{\text{ビット}}{\text{トラック}}\div 32\frac{\text{ビット}}{\text{語}}=128\frac{\text{語}}{\text{トラック}}$$

(a)

(b)

(c)

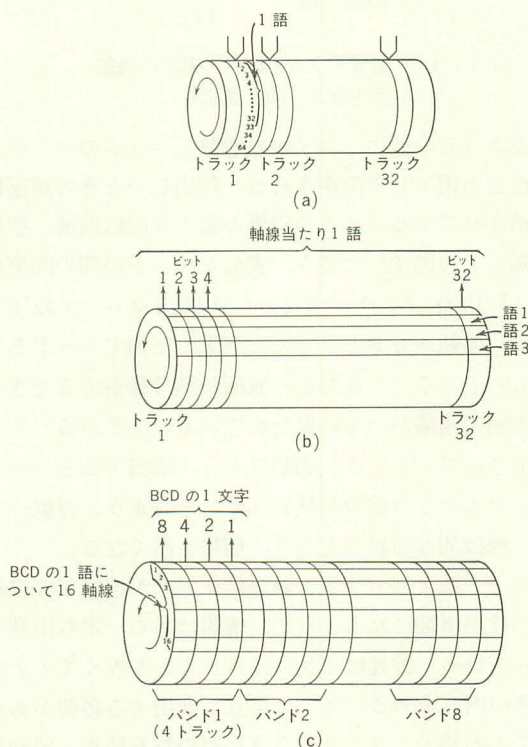



図 11-16　磁気ドラム上のデータの構成：(a) ビット直列に構成された記憶，(b) ビット並列に構成された記憶，(c) ビット直並列に構成された記憶。

したがって，トラック 1 には 0～127 番地，トラック 2 には 128～255 番地，…と入っていることになる。各トラックは 1 つ 1 つヘッドがあるから，呼出し時間はどのトラックも同じである。一度要求された語のはじめがそのトラックの固定されたヘッドの下に現われると，その語全部を読み出すのに 32 ビット時間かかる。もし，ドラムの回転速度が毎分 3600 回転（1 秒間に 60 回転）だとすると，1 秒間の転送量は，

$$128\frac{語}{回転}\times 60\frac{回転}{秒}=7,680\frac{語}{秒}$$

である。

ドラムのデータ構成の第2番目のものは，ビット並列（または語直列）である。この場合，1 語中の 32 ビットは同時に読み込んでしまう（図 11-16b)。アクセスは逐次的であるが，1 つ1つのビットはべつべつのトラックの上にあり，1語は1ビット時間で得られる。この場合，1 秒間に転送される語の数（これを**転送率**という）は，

$$4096\frac{語}{回転}\times 60\frac{回転}{秒}=245,760\frac{語}{秒}$$

である。これは，ビット直列の場合よりはるかに高い率であり，目的によっては望ましい方式である。

もう1つの方式は上の 2 つの混合で，ビット直並列の方式である（図 11-16c)。これは，特殊なコードのデータに対し有用である。たとえば，2進化10進数(BCD)のデータを扱うときは，1 文字を蓄えるのに4ビット必要である。このことから，ドラムを横に8つのバンドにわけ，1 バンドには4つのトラックを入れることにする。1 バンドの4トラックを並列に使って，BCD の1文字4ビットを表わす。16 文字を1語とすると，1 つのバンドに 256 語入る。前と同じ計算をすると，転送率は 15,360 語/秒となる。以上のすべての場合に，ビットを対象にすると，

$$4096\frac{ビット}{回転}\times 60\frac{回転}{秒}=245,760\frac{ビット}{秒}$$

である。これは約 250 kHz の速さで動作する回路を必要とする。

最大の呼出し時間は1回転すなわち 16 ミリ秒 (1/60秒/回転) であり，平均呼出し時間は，8 ミリ秒である。必要な情報を得るにはかなり長い時間かかるが，データはドラムから高速で得られていることは明白である。最初の呼出し時間とデータ転送率(語/秒)の両方がドラム記憶装置を述べるのに必要である。

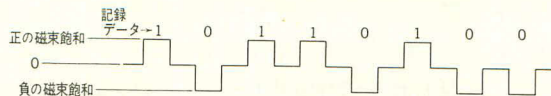



図 11-17 RZ 方式で記録されたデータ

磁気ドラムへの記録 磁気ドラムの一般的な構成を考えたので，今度は，データがどのようにドラムの表面に蓄えられるか，また，どのようにして読んだらよいかを見ることにする。記録の方法には，RZ (return-to-zero, もっと一般

的には return-to-reference）と NRZ（nonreturn-to-zero）の2つの方法がある。**RZ 方式**では，1つのトラックの上に磁束を記録するのに，隣合った情報ビットの間で1回ごとにある基準値に戻されるのである。普通，この値は0である。この0の磁束に対して，ドラムの表面は2方向のうちどちらかの方向に飽和した磁束を保っていることになる。図 11-17 に RZ 方式の入力データの例を示してある。ビットが格納されている表面を飽和させておくと，書込み電流の変動によって起こる電圧変動が少なくなり，読出しが容易になる。RZ 記録用ヘッドが図 11-18 に示してあるが，これは ONE と ZERO を記録する例である。

普通，情報密度は，ヘッドと表面の間隔，表面を通る磁束の広がりなどに関

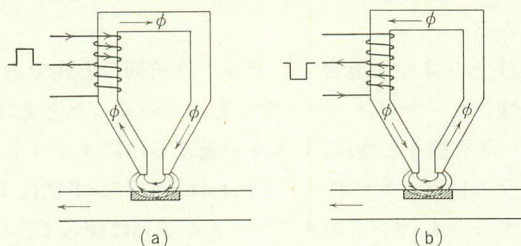



図 11-18 RZ 方式を用いた磁気記録

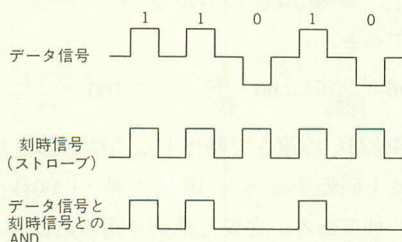



図 11-19 RZ 記録からのデータ読出し

係する。これはまた記録の方法にも関係がある。RZ 方式では，磁束0の安全帯が隣合ったビットの間にあり，さらにデータを読むとき，負の電圧については考える必要がない。図 11-19 は，いくつかのデータ パルス，ストローブ（strobe）パルスとして使われる刻時信号（データが読まれる特別な時間を指定するのに用いられる）や出力データの関係を示したものである。

AND ゲートを使っていて，刻時信号とデータ信号の両方があるときだけ出力がでるようにしてあるので，負のデータ パルスがなくても，同じパルスが得られることに注意しよう。こう考えると，同じ場所に，より多くのデータが

蓄えられるのではないかという気を起こさせてしまう。この密度を増加させた記録法を考える前に，**RB** (return-to-bias または return-to-reference) **方式**が，RZ と同じ記録密度を持ちながら，重要な利点を持っていることを指摘しよう。もしバイアスの大きさが，ONE の信号と反対の方向の飽和点になっていたとすると，再生された信号は，RZ のものの大体2倍である。そして，読出し増幅器で両極信号を扱う必要はない。図 11-20 は RB 方式によって読まれたデータを示したものである。

RZ より高い密度でデータを記録する方法に **NRZ 方式**がある。図 11-21 に図11-17, 11-20 と同じデータによる例を示してある。この3つの図の違いを注意深く調べてみよう。特に，NRZ 方式と RZ 方式の相異に注意すると，この2つはちょっと見れば同じように見えるが，注意深く見るとまったく異なっているのである。NRZ の記録法だと，2 進数は，磁束のレベル（飽和磁束が望ましい）で表わされている。NRZ 方式で同じ数字が続いて記録されると，磁束

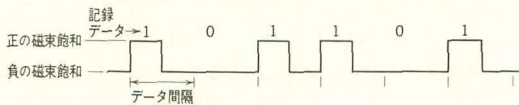



図 11-20　RB 方式で記録されたデータ

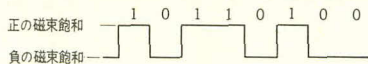



図 11-21　NRZ 記録方式を用いて記録されたデータ

は変化しないままになっている。しかし，RZ や RB 方式ではたとえ同じ2進の値であっても，隣接したビットの間で常に磁束は変化している。NRZ 方式での磁束変化の数は，ONE と ZERO が交互に記録されているときに最大である。それでも RZ 記録法による磁束の変化の回数の半分である。NRZ を使うと，RZ よりも記録密度は高く，2 倍である。NRZ では，データの変化があるときだけ磁束変化が起こり，ビットが同じときには何も起こらないから，NRZ のデータを読むときは，隣接したビットが同じ場合に読んだものが何かをはっきりさせるために，なんらかの手段を講じなくてはならない。図 11-22

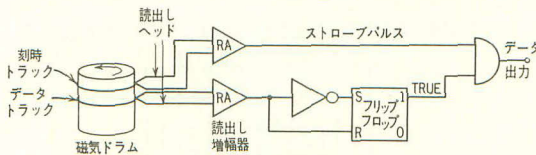



図 11-22　NRZ の記録データを読み出す回路

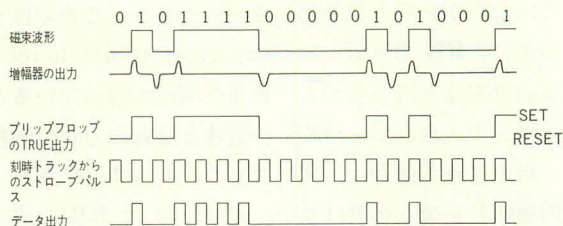



図 11-23　NRZ の記録データを読み出すときの信号の波形

は NRZ で蓄えられたデータを読み出すのに使われる簡単なフリップフロップの回路である。この回路からの信号の波形を図 11-23 に示す。

NRZ で記録されたデータの磁束の波形は，格納させた情報をはっきりと示している。磁束の変化だけが検出されるので，ONE や ZERO が反復して現われて，一定の磁束が続くとき，それは増幅器の出力波形には直接現われない。この出力がフリップフロップの SET や RESET に使われると，記録されたビットが明白になる。先に示した回路では，負方向の信号がフリップフロップ入力をトリガしている。ドラムの刻時トラックからとり出したストローブすなわち刻時信号を使って，データ出力が作られ，ビット直列のデータが作り出されて，記憶装置の外で用いるため，バッファまたは格納レジスタに読み込まれる。

もう1つの NRZ 記録法は，2 進記号の1つだけを記録するものである。この記録法は **NRZI 方式** (I は invert) と呼ばれ，2 進の ONE のときだけを考え，それらを1つの磁束変化として記録する (図 11-24 参照)。このため，1つの数字がときには正の記録信号になり，ときには負の記録信号にもなる。磁束の変化があるときには，ONE が格納されていることを示し，磁束変化が何もないときには，ZERO の意味である。図 11-24 (b) はONE が続いている

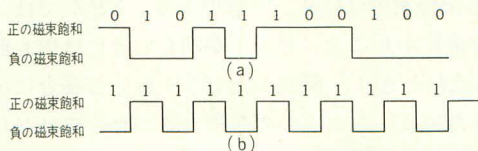



図 11-24　NRZI 記録法を用いたときの2進数波形

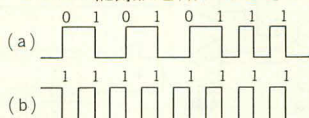



図 11-25　NRZ 記録法のデータの位相記録

とき，ONE のたびごとに磁束の変化があることを示している。このデータもいま述べたものと同じ方法で読まれる。

NRZ の第3の方法は，**位相記録**（“Ferranti” あるいは “Manchester” 方式ともいう）と呼ばれているものである。図 11-25 にこの方法の 2 つの波形を示す。たとえば，ONE を記録するときは，そのビットは正の磁束から負の磁束への変化として記録される。ZERO のときはその反対に負の磁束から正の磁束への変化として記録する。この方法で一番重要なことは，この波の周波数の最大（全部 ONE か全部 ZERO のとき）と最小の比が 2：1 となることである。最大周波数が 200 kHz なら，最低は 100 kHz となる。最小の周波数は，ONE と ZERO が交互に出てくるときである。ほかの2つの NRZ 方式では，たくさんの ONE や ZERO が連続した記録は直流動作に近いような定常的な，あるいは静止した磁束で現われる。これは書込み，読出しのための増幅器が直流から最大周波数までにわたって動作しなくてはならないので，それに1種の制約が課せられることになる。先に述べた位相記録方法では，周波数の幅は完全に限定されていて，変圧器結合増幅器を用いることができる。格納した情報を読む操作は本質的には，最初の NRZ で述べたものと同じで，ここで詳しく述べる必要はない。必要なのは，他の2つと同じように，データの記録や読出しをタイミングの信号に注意深く合わせることだけである。

いままでデータが磁気ドラムにどのように格納されるかとか，ドラムでどのように構成されているかを考えてきたが，今度はドラムの読み・書きの働きが全体としてどのように制御されるかを見ることにしよう。円周上の適当な出発点にビットが1つだけある**出発点トラック**（origin track）がある。もう1つのトラックが同期をとるために用意されていて，**刻時トラック**（clock track）と呼ばれている。このトラックには，ONE と ZERO が交互に入っていて，刻時信号を出す。この周波数はドラムの回転速度（インチあたりビット数）によって定まる。この速度はかわることがあるし，外部の刻時信号とドラムに記録された刻時との同期がとれないこと，すなわちドリフトしたりすることがあるので，このドラムで制御された時計が，読み・書き動作の同期をとるのに一番よい。ドラムの外部の制御装置には，トラック刻時信号を使い，ただ1個の出発点パルスによってリセット，すなわち計数を開始し始めるビットあるいは語計数器がある。このようにデータは常にトラック レイアウト*によって時間を合わされている。機械的な原因でトラック間にごくわずかの不整がある場合，

*〔訳注〕トラックに記録された刻時信号の模様。

データは各ビットに都合のよい一時点でストローブ，すなわち読み出される。タイミングやストローブの詳細については，磁気ドラム関係のたくさんの本*があるからそれらを参照されたい。ここで大切なことは，ドラムに格納されたデータは周期的に扱われなければならないこと，また，この記憶装置を働かせるのに，ドラム上の出発点トラックや，刻時トラックと関連して働く磁気ドラムの外部回路が用いられることである。図 11-26 はこの周期的な読出しを示すブロック図である。

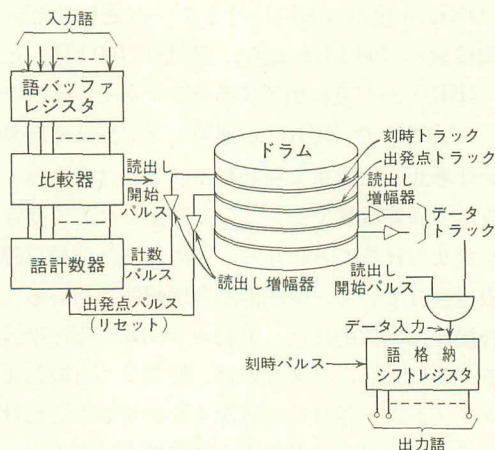



図 11-26 周期的な読出しゲート

算術演算部で連絡をとるのが A レジスタであったが，語バッファ レジスタは，計算機と記憶装置の間の連絡をとるものである。命令は記憶装置に対してどの語が必要なのかを指示する。出発点トラックからの信号がくると，計数器は 0 になり，ドラムの刻時トラックからのパルスを数える。比較器で 2 つのレジスタ**に同じ番号があることがわかると，読出し開始信号を出す。これは読出し増幅器からきたデータを 格納シフトレジスタ*** に転送する。刻時パルスは，データと同期しているので，桁移動のためのパルスとしても使われる。10 個のパルス（ここでの仮りの語の長さ）の読出しの後，読出し動作は終り，そのデータは計算機の他の回路で利用できるようになる。

---

* 〔訳注〕 たとえば，情報処理学会編「電子計算機ハンドブック」。

** 〔訳注〕 語 バッファ レジスタ と語計数器

*** 〔訳注〕 格納シフトレジスタ（storage shift register）は計算機と記憶装置との間にある情報 1 語を入れる バッファ レジスタでシフトの機能をもつ。

11-4　その他の記憶装置

これまでの2つの記憶装置はいずれもデータを書き込むと，前にあったものは消えてしまう，いわゆる消去可能な記憶装置である。しかしほかのいくつかの装置はこの望ましい消去可能の特徴がない。光学的記憶装置はこの消去不可能な記憶の一例である。光学的に透明な材料でできているドラムに感光乳剤を塗り，決まったコードまたは文字図形を焼きつける。一度データがドラムに書き込まれると永久に消えることはない。この型の記憶はある種のデータには適している。計算機が使うある種の表はこのような記憶装置に格納することができる。呼出し時間は，ドラム記憶装置なので，機械的スピードで制約され，平均1ミリ秒位がちょうどよいとされている。光学記憶装置の2つの主な特徴は，その記憶容量が大きいという点と，磁気ドラム記憶装置より安くあがるという点である。データの読取りは，光源からデータ トラック を照らし光学検出器を用いることによって行なわるれ。ONE は光学ドラム上の被覆のないことで表わされ，光学検出器に光があたるようになっている。また，ZERO は不透明な被覆があることで表わされ，光はあたらない。データは磁気ドラムでのようにトラックに格納され，逐次的に読まれる。刻時トラックや出発点トラックも読出しの際の同期のためにドラムの表面に印刷されている。

フェライト シートとか**薄膜記憶装置**というのは原理的にはコア記憶装置と同じであるが，構造上の違いから高速，小型，低価格という特徴が生まれてくる。これらはとても魅力的な特徴なので，近い将来この改良されたコア記憶装置がコアにとってかわるだろうと思われる。いまのところは，まったくの新製品で一部に使われているにすぎない。この装置は，平版の上に細い銅線を糸状におくか，またはもっと最近では平版上にこの導線を埋め込むようにして作られている。そのためコアを1つ1つ独立におくよりも構造が緻密となり，小型であるために，より高速の動作ができ，製作が簡単なために，より安価である。構成はコアとまったく同じであり，もちろん，ランダム呼出しができる。

最近話題になっているもう1つの磁気記憶装置は**ツイスタ記憶装置**である。ツイスタとは，その名の示す通り，薄い磁性のリボンを絶縁銅線に巻いたものである。リボンはとても薄く，約 1/2 ミル（12.7ミクロン）で，針金は 1000 分の 2～3 インチ位（50～76ミクロン）の太さである。銅線上のリボンの巻き方はちょうど床屋のしるしである柱の模様を思い出せばよい。もう1本の線が

一群のツイスタのまわりに巻いてあって，この線と銅線を通る電流パルスが一致すると，リボンの特定の部分が磁化される。ツイスタの縦方向にたくさんの線を巻くことにより，データをリボンの縦方向に格納できる。ツイスタ記憶装置にはこのほかに 2,3 の形があるので，詳細は文献*を参照されたい。ツイスタは，ある場合には一時記憶に，ある場合には永久記憶に使われる。現在のところ，コアよりも容量が小さく，速度も遅い。ツイスタ素子がますます使用されるかどうかいますぐ明らかになるものではないが，時間をかけて研究をすれば将来の応用が決まることになろう。

現在使われていて最も速い記憶装置は，記憶素子として**トンネル ダイオード**を使ったものである。トンネル ダイオードは，2 端子の半導体で，2 安定状態をとりうる。図 11-27 に，その V-I 特性を示してあるが，この負荷線が 2 つの安定状態を示している。素子はキロメガヘルツの速度で理想的に動作されうる。その結果，1 ナノ秒（10^{-9} 秒）の呼出し時間で働くことになる。いまのところ動作速度を制限しているのは，実際の回路の結線，配線の浮遊容量などである。トンネル ダイオード記憶装置は，高価であること，記憶容量が小さいこと（1 つのトンネル ダイオードにつき 1 ビット），まだ引き続き研究の必要があることなどの理由から，実験的な装置に使われているにすぎない。さらに研究が進めば，この記憶素子の使用範囲も決まるだろう。

ほかにもまだ多くのものがあって，どれも重要ではあるが，ここでは主な記憶素子の特徴を概観してきた。大きさ，価格，速さ，扱いやすさ，記憶容量などを考え合わせるとどの形も適当ではなくなってしまうので，この分野はこれからも絶えず研究され，かえられていくことになろう。

---

*〔訳注〕 1. Bobeck, A. H. 'A New Storage Element Suitable for Large-sized Memory Arrays——The Twistor', Bell System Technical Jour., **30**, pp. 1319-40 (1957).

2. Fisher, R. F. and Mallery, P. 'Counter Wrapped Twistor', Proc. Electronic Components Conference, Washington D. C., May 1960, pp. 129-33.

3. Schwaltz, S. J. and Sallo, J. S. 'Electro-deposited Twistor and Bit Wire Components', Trans. I. R. E. on Electronic Computers, **EC-8**, pp. 465-9 (1959).

4. Looney, D. H. 'A Twistor Matrix Memory for Semipermanent Information', Proc. Western Joint Computer Conference, March, 1959, pp. 36-41.

5. Barrett, W. A., Humphrey, F. B., Ruff, J. A. and Stadler, H. L. 'A Card-changeable Permanent-magnet Twister Memory of Large Capacity', Trans. I. R.E. on Electronic Computers, **EC-10**, pp. 451-61 (1961).

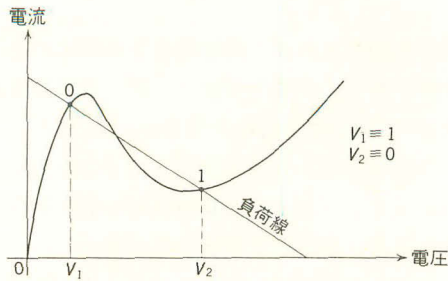



図 11-27 トンネルダイオードの特性

要 約

計算機の記憶装置は，入出力関係と区別して考えると，小さくて，比較的安くなければならないが，できることなら，とりわけ速いことが必要である。磁気コア記憶装置は，現在，最もよく使われており，その応用はますます増加していくであろう。コア記憶装置はより小さく，より速くなってきたし，容量も次第に大きくなってきている。数年前には，32,000 語程度のコアが大きいものだったが，いまや，数十万語位の容量はまれではない。呼出し時間（記憶装置から語を読み出す時間）は現在のコアでは1マイクロ秒の 10 分の1程度である。

コア記憶装置は電流一致を用いたランダム呼出しである。この方法はまた，電線の節約にもなる。ランダム呼出しというのは，対象となる記憶場所から語をとり出すのにどこからでも同じくらいわずかな時間ですむような方法である。実際のコアからの読出しは破壊的であるが，読み・書きのサイクルを使って，読まれたデータをもとに蓄え直すことができる。もちろん，読み・書きサイクルの書込み動作のとき，新しいデータを必要な記憶番地に置くことができる。

磁気ドラム記憶装置は，コアが出現してからほどなく内部記憶装置としてはあまり使用されなくなったが，速度が問題ではなく，価格が安いほうがよいという場合にはまだ一般に使用されている。平均呼出し時間は速いもので1ミリ秒ぐらいであるが，1つのヘッドで1トラックから数千ビットも得ることができ，容量のわりに安価であることが大切なところである。

ドラム（テープまたはディスク）へのデータの記憶法にはいろいろある。RZ, RB 法というのはビットごとに変化が2度起こるので，読出しには容易な方法である。NRZ 記録法は同じビットが続くと波形に変化が起こらないから，任意の時点で読み出すときは，いくらか多くの注意を払わなくてはならな

い。もっと複雑な回路を用意すれば，記録密度を2倍にできる。NRZI 法は，RZ の2倍の記録密度がある。位相記録法は，最小周波数と最大周波数の比が2対1で，変圧器結合*の増幅器で読み・書きができる。

磁気ドラムのデータ構造には，ビット直列，ビット並列などがある。どの場合でも，ドラムには，ドラム（および計算機組織の他の部分）のタイミングをとるための刻時トラックと，現在，どのデータが読出しヘッドを通ったか，データをいつ書込みヘッドから送り込むのかを知らせる信号のある出発点トラックがある。

問　題

1. 4096 個のコアの板の行番地レジスタ X と列番地レジスタ Y に，何段の記憶装置が必要ですか。

2. 1語 20 ビットであるとき，10,240 個のコア記憶装置に対して，行番地レジスタ X，列番地レジスタ Y および語レジスタ Z に，それぞれ何段の記憶装置が必要ですか。

3. 1語 10 ビットを持つ大きさ 20,480 のコア記憶装置に，X, Y および Z レジスタのため，何段のものが必要ですか。

4. (a) 1分 18,000 回転するドラム記憶装置の平均呼出し時間はいくらですか。

(b) 1トラックに 1024 ビットがあるとき，ビット転送率（ビット/秒）はいくつですか。

(c) 1ビットが読まれる間の時間間隔はいくらですか。

5. 16 トラックあり，1トラックに 2048 ビット持つドラムがある。もしそのドラムが1分 12,000 回転するならば，平均呼出し時間はどうなりますか。

6. (a) 問題5において，ドラムがビット直列であるとすると，データ転送率はいくらですか。

(b) 同じく，もしビット並列であるならば，データ転送率はいくらですか。

7. 問題5において，ドラムを BCD 文字に使うための2通りの方法を示しなさい。

8. RZ 記録法で，次のデータを記録したときの磁束波形を書きなさい。

(a) 1101101110110

(b) 1010110011000

9. (a) 問題8において，NRZ 法だと磁束波形はどうなりますか。

(b) また NRZI 記録法の場合について考えなさい。

10. 問題8のデータについて，位相記録法の場合の磁束波形を書きなさい。

---

*〔訳注〕 信号を増幅器へ入れるとき，不必要な直流成分を取り除く必要が生じるならば，CR の組合せや，変圧器によってこれを行なうことが多い。これは，信号が脈流のとき，すなわち，ある周波数をもった交番電流成分をもっているときのみに使用可能である。

1100＝12

入出力装置

12-1　入出力技術一般

たいていの電子計算機には，データを計算機に与えたり計算機から読んだりすることができる装置がついている。この章ではその入出力装置を主題にして考えよう。計算機の使用者にとって，装置の入出力部分は非常に重要である。使用者は，それらを用いてデータを最も効率的に，そして迅速に取り扱うことができるようになる。電子計算機本体は，多くの問題を高速で解く計算機にすぎない。一般に，使用者に最も関心があるのは，入出力装置である。簡単に言えば，入力装置は電子計算機の中で使う2進または2進符号化データを作る機器からできている。出力装置は外部での処理または貯蔵のために，電子計算機から2進数（または2進符号化データ）を受け入れる機器からできている。穿孔カード，磁気テープ，磁気ディスク，紙テープ，高速印刷機など非常によく知られている装置は，いろいろな形式や組合せで汎用計算機に使われる。特殊目的の計算機は，アナログ-ディジタル，または，ディジタル-アナログ変換のために，標準品ではあるが一般にはあまりよく知られていない装置を使う。それらは違った考え方と手法を使うので別に議論する。

入出力装置の制御は，中央演算制御装置または付属する特殊装置，またはその両方で行なわれる。1つまたは2つの入出力装置が働いているときは，多分，電子計算機はいつデータを入れ，どのデータを使うか，いつ止めるかを決定する信号を出して完全に制御するであろう。多くの装置を制御する場合，電子計算機が特定の外部装置に始動を伝える刻時信号を送り，他のすべての制御を外部装置にまかせてしまうこともある。入出力装置は計算機システムの中で，最も遅いものであるから，連続的に演算するために計算機の算術演算装置に十分なデータを供給するため，多くの種類の装置を結合する必要がある。1つの計

算機本体に 10 数本の磁気テープや数本の紙テープ装置を備えて，データを他の磁気テープや高速印刷装置に書き出している間に入力データを与えることが可能である。事実，データを十分速く読み込んだり，また目で見える表示装置にデータを書き出す問題は計算された大量のデータを使用者が使用できる形にするため，タイプ印刷，グラフ表示，オシロスコープ，陰極線表示などの分野の絶えざる改良を必要としている。

一方，アナログ-ディジタル（A/D）とディジタル-アナログ（D/A）変換装置は一般にデータを十分速く電子計算機に与えることができる。しかし，変換精度については改良を続ける必要がある。直流や交流の電流および機械的な軸の回転はアナログ信号の例であって，電子計算機で扱うためには 2 進数に変換しなくてはならない。たとえば，0V から +10V の間の直流電圧を 0.5V の段階に分け，各 0.5V きざみのものを，0 から 20 までの 2 進数に対応させることができる。2 つの直流電圧の和は，これらの数量化された 2 進数を計算機内部で加え合わせることによって求められる。そして，もし必要なら，結果の数は電圧に変換し直せばよい。明らかにこのような単純な演算にはこの手順は必要ない。しかしながら，高速・高精度の多くの演算が必要となると電子計算機は重要になる。直流電圧からディジタル量へ（ディジタル量から直流電圧へ）とか，軸の位置の A/D と D/A のようないくつかの重要な特定の変換については，いくらか詳しく述べる。

12-2 穿孔カード

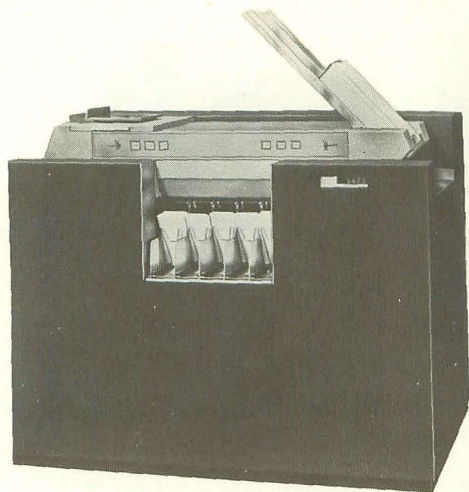
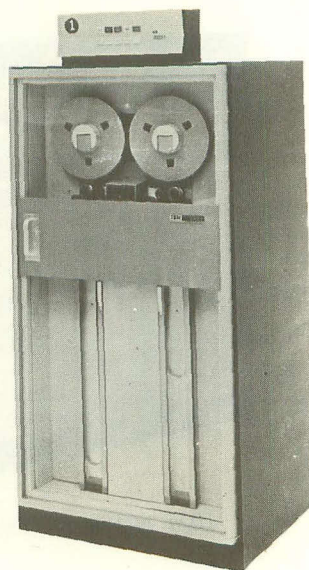
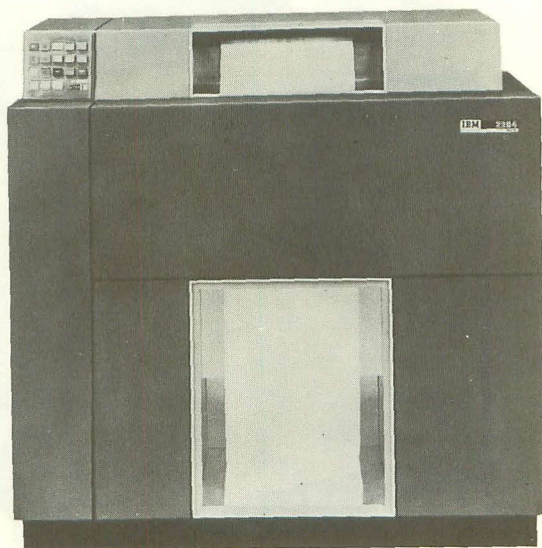
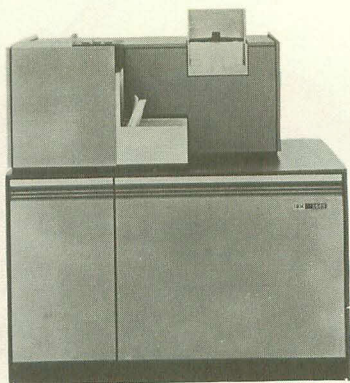
穿孔カードによるデータ処理は，計算機にディジタルの情報を与えるために広く使われている方法である。カードは一定の形状をしていて 1 と 0 に対し，それぞれ穿孔する穿孔しないを対応させて，2 進データを蓄える。もちろん，データの貯蔵は永久的である。多くの操作にとってこれは大変望ましい特徴である。1 枚のカードには，一般に特定の項目について少量のデータを蓄えることができる。1 枚のカードは 1 つの項目（個々の学生とか工場の 1 つの製品，たとえば自動車部品）と直接の対応を持っているので，各カードすなわち項目ごとに取り扱いたいときには便利である。1 組のデータが 1 枚のカードに蓄えられているときは，カード分類機を使うことにより，カードを分類することができる。学校の各学生に対する 1 組のカードはクラス別，学年別，またはその他の目的に従って分類することができる。

IBM 1402 カード読取り穿孔機

IBM 2401
磁気テープ装置

IBM 2204 印刷機

図 12-1　典型的な入出力装置

IBM 2671 紙テープ読取り機

IBM 1442 カード穿孔機

IBM 2311 ディスク記憶装置

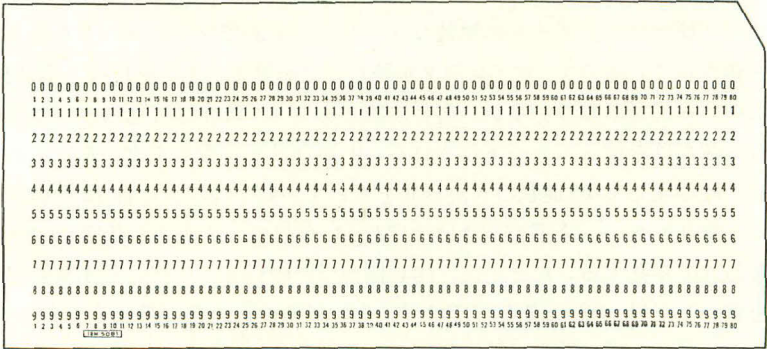



図 12-2 標準 80 欄カード

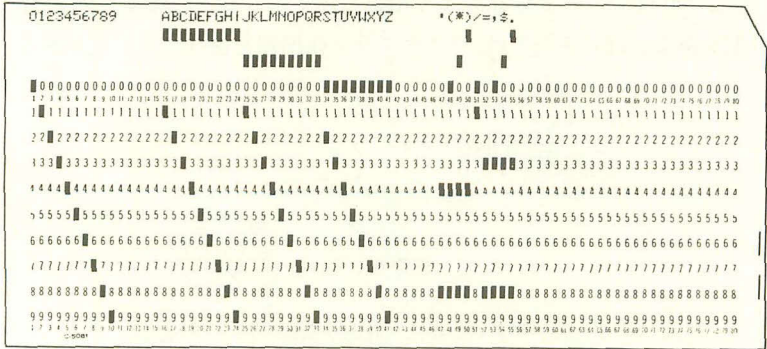



図 12-3 英数字コードを示す穿孔カード

表 12-1 穿孔カードのコード

数字だけ	“12” と数字	“11” と数字	“0” と数字
0 = 0			
1 = 1	1 = A	1 = J	
2 = 2	2 = B	2 = K	2 = S
3 = 3	3 = C	3 = L	3 = T
4 = 4	4 = D	4 = M	4 = U
5 = 5	5 = E	5 = N	5 = V
6 = 6	6 = F	6 = O	6 = W
7 = 7	7 = G	7 = P	7 = X
8 = 8	8 = H	8 = Q	8 = Y
9 = 9	9 = I	9 = R	9 = Z

カードは 80 欄，12 行から成っている（図 12-2)。各欄は 1 つの文字を表わすために使われる。データは単純な（しかし効率的でない）コードを使ってコード化されているので，オペレータが読むのは容易である。表 12-1 と図 12-3 に，このコードを示してあるが，容易に読めることがわかるであろう。数字はその行の数だけを使ってコード化される。0 から 9 まで数字を書いた行があるが，そこに穿孔して 10 進数の 0 から 9 を表わす。数字の場合には 12 行のうち 1 つしか穿孔しない。文字のときには 12, 11, 0 行のいずれかを数字 (1-9) の行と合わせて穿孔する。たとえば，同じ欄の 12 の行と 1 の行に孔があれば文字Aを表わし，11 の行と 2 の行では K, そして 0 の行と 9 の行は Z を表わす。

穿孔されたカードは非常に速く処理され，現在 1 分間 1000 枚以上も読める機械が使用されている。穿孔する速さは必然的に遅く，1 分間 120 枚（1 秒に 2 枚）から 250 枚ぐらいである。よく用いられているカード読取り装置には，2 つの型がある。遅い方法では，ワイヤブラシ検出器を使う。もし孔があると，

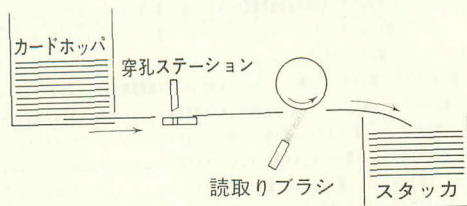



図 12-4 ワイヤブラシ穿孔・読取り装置（IBM 社提供）

ブラシは金属板と接触し，2 進数の 1 の信号が読み込まれたことを知らせる。ブラシ型の読取り機はせいぜい 1 分間 200 枚までしか処理できないが，より速くカードを読み取るためには光電読取り機を使う。カードの一方の側にある光源はもしカードに孔があれば別の側にある光検知器（フォトダイオード，フォトトランジスタ，光電池，その他）を刺激する。もしカードに孔がなければ，何の影響も与えない。光電読取り部分の反応は速いので，読み取り速度はかなり速くすることができる。しかし，穿孔するということは，カードから物質をとることであるから，当然遅くなる。図 12-4 にワイヤブラシ型の穿孔・読取り機を示す。

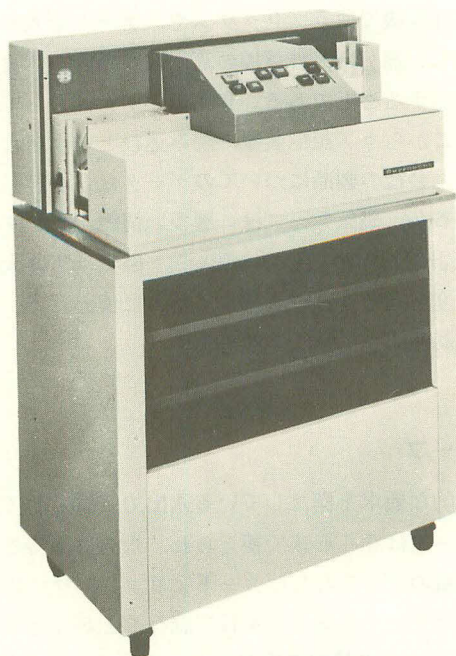
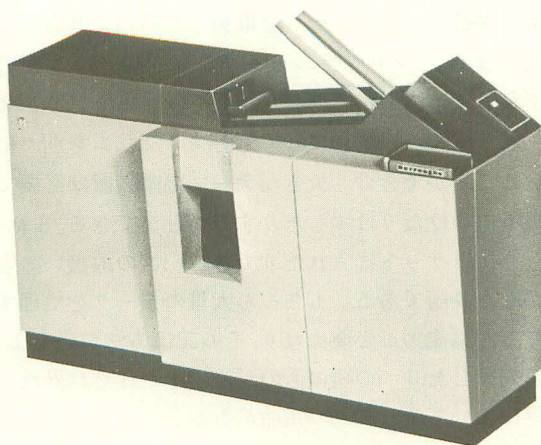
カード読取り機やカード穿孔機の制御は，一般に別の装置で行なわれる。電子計算機本体は，いつ読取りを始めるか命令するだけである。多くのカードに蓄えられたデータは，コア記憶装置の一部に格納するために使われる場合もある。一度入れられると，計算機は内部記憶装置を使い，高速で演算を行なう。

（a）カード読取り装置

（b）カード穿孔装置

図 12-5　高速カード読取り・穿孔装置

穿孔カードには，計算機で使うプログラムや，データを入れることができる。仕事の一番むずかしい部分は，計算機の算術演算装置を休ませないように，十分速くデータを供給することである。穿孔カードを前もって磁気テープに読み込んでおいて，そこからコア記憶装置に読み込むという組合せをとることがある。穿孔カードは，会社の製品についてのデータなどを容易に取り扱えるという点から必要であるが，磁気テープは一層多い情報を1ブロックにまとめて使うことができる（計算機への読込みも速い）。磁気コアは大量のデータの一部を計算機の演算速度と同じ速度で処理するために使われる。図 12-5 に高速カード読取り装置と高速カード穿孔装置を示す。

12-3 穿孔紙テープ

もう1つの実用的な要求を満たしている入出力装置として穿孔紙テープ入出力装置がある。データは穿孔形式で蓄えられ，永久に保持される。穿孔テープの主な長所は，少量のデータならばカードよりコンパクトに蓄えることができ，テープにどんなデータがあるかを目で読むことができる（磁気テープはできない）。したがって少量の情報の取扱いがしやすく，また安価であることである。磁気テープの場合にはかなり大きいリールをつけて取扱うし，少量の情報を処理するようには作られていない。穿孔カードのように1枚1枚分けて扱う必要がなければ，穿孔テープはデータを取扱うためのより安価なより効率的な方法である。たとえば，プログラム ルーチン を穿孔テープに蓄えることもある。あるプログラムでは，テープの長さが5フィートであり，またほかのでは6インチであったりする。いずれにしても各プログラムを別々に蓄え，必要なとき機械に入れることができる。大きなテープ駆動装置は必要ではないし，テープは容易に機械につけたりはずしたりすることができる。また手に持ったり，目で見て調べたり，ポケットに入れたりできる。この取扱いの容易さは磁気テープと比べて顕著な長所である。もちろん大量のデータを処理するときには磁気テープの速度と記憶能力が必要になり，その記憶方法が最もよく用いられる。重要なことは，考察した3つの型のものが操作上それぞれ異なった長所を持っていて，産業界においてそれぞれの用途があるということである。

テープ読取り装置 （p.232） は，機械的読取りで，1 秒間に約 100 文字，光電読取りで 1000 文字である。テープ穿孔装置は1秒 50 文字の速度で，1インチに 10 文字ずつ穿孔する。一般に，テープは2進化情報のための8単位と，

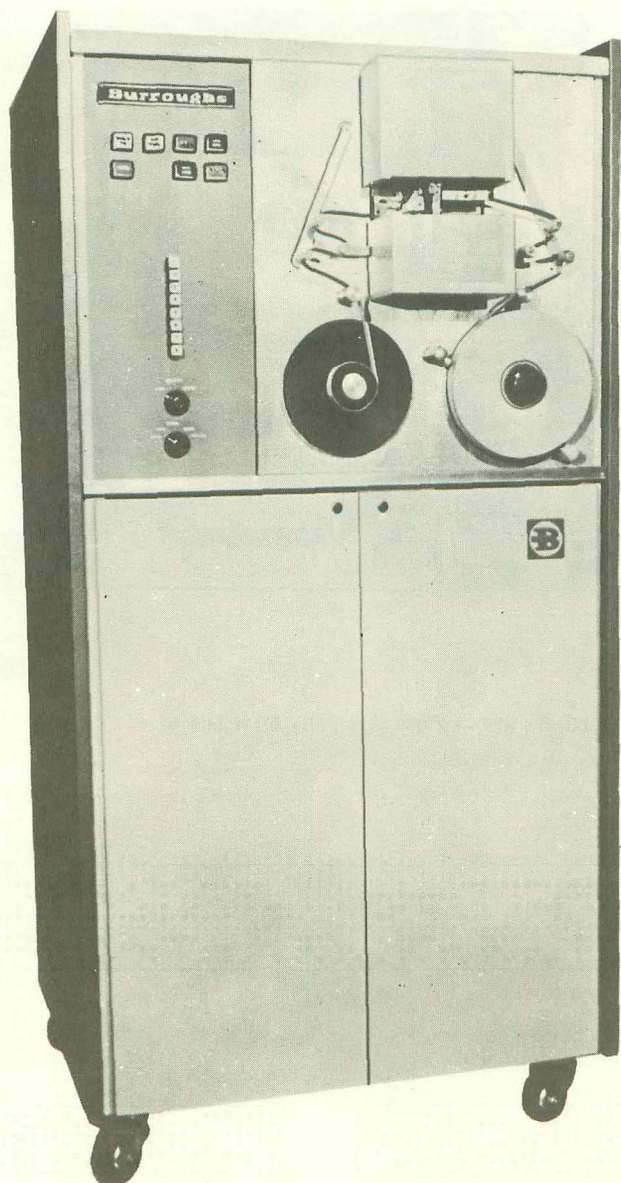



図 12-6 紙テープ穿孔装置：(a) Burroughs B341 紙テープ穿孔機

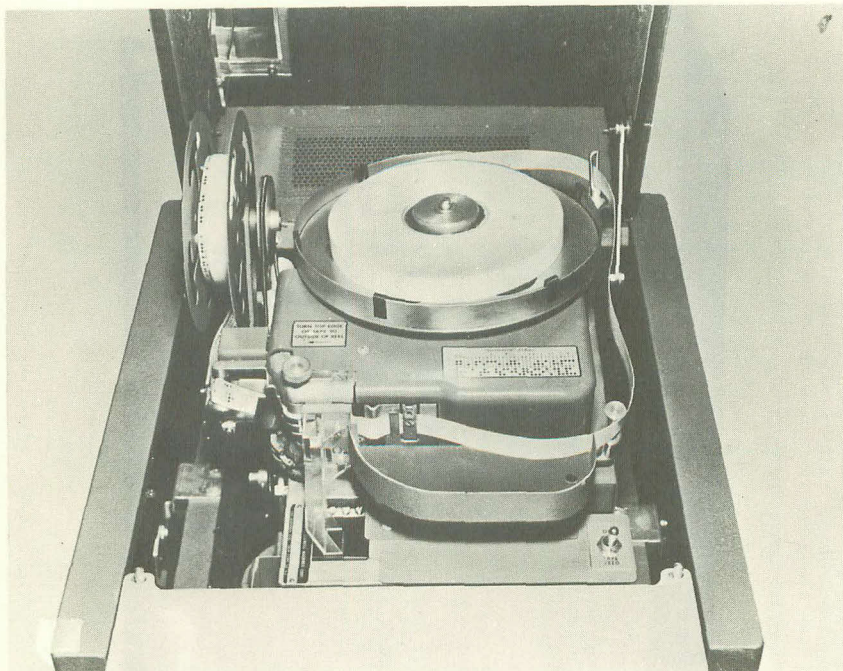



図 12-6　紙テープ穿孔装置：(b) IBM 962 紙テープ穿孔機

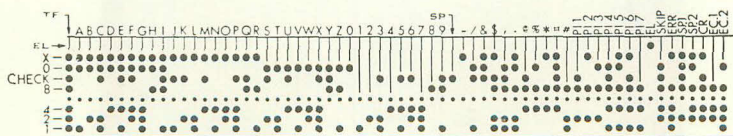
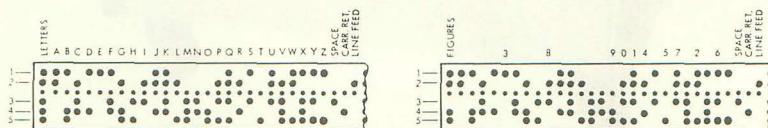



図 12-7　紙テープ　(a) 8チャンネルコード，(b) 5チャンネルコード (IBM 社提供)

テープ送りのためのスプロケット用の孔がある (図 12-7)。8 単位により 8 ビットコードを表わして，各列に 1 文字が入る。テープはプログラムに応じて，長くても，短くてもよい。テープを読むときは，読取り装置がスプロケットの穴を使ってテープを送り，ワイヤ ブラシ ピン または光電読取り機が使われる (図 12-7c)。

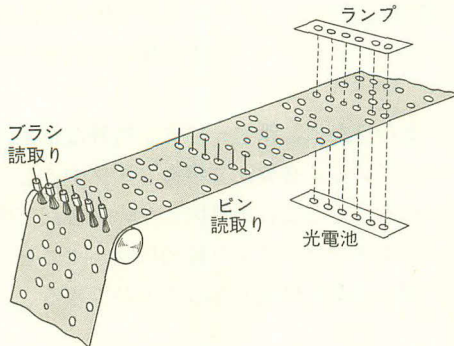



図 12-7(c) 穿孔テープの読取り

データは 1 列に 1 文字穿孔される。普通スプロケットの片側に 3 行，残りは反対側にある。その 3 行は図 12-7 に示すように 1, 2, 4 と重みのついた場所である。スプロケット孔の次の行は重み 8 の位置である。5 単位コードでは，32(2^5) 文字しか扱えないので，扱える文字を倍にするために特別なシフト機構が使われる。そのようなシフト機構の 1 つとして，文字の前に特別なコードを置くことがある。特定のコードがあると，以下に続くものは文字であることを示し，別のあるコードならばそれに続くものが数字，または特殊文字であることを示す。6, 7 または 8 単位コードの場合にはそのような問題はない。というのは，64(2^6) またはそれ以上の文字を表わすことが可能であり，タイプライタの鍵盤にある数字と英字と特殊文字の数は，普通 64 以下であるからである。5 単位コードはテレタイプ装置でよく使われる。それは 1 文字のために送信するビットの数が少ない方が好ましいからである (より高単位のコードがやがて使われるようになるだろうが)。

チェック ビット の場所には，データを読む際，正しいかどうか検査するための パリティ ビット が入る (4 章で論じたように通常奇数パリティ)。テープのスプロケット孔には 2 つの重要な目的がある。第一に，それは読取り装置を通してテープを動かすものである。テープを進める機械的な装置はスプロケット孔を用いて一度に 1 文字ずつ動かす。第二に，スプロケット孔を検知して読取り機構 (ブラシまたは光電池など) に文字が正しく位置していることを読取り

装置に伝える読取りパルスが出される。スプロケット孔はデータの孔より小さく，テープの中央付近に位置している。スプロケット孔が読取りブラシの場所にあるときにだけデータは読まれる。というのは，孔が読取りブラシの下にある間，信号パルスが生じるので，スプロケットパルスはより狭いパルスでデータパルスの中央に位置している。小さいスプロケット孔は読取り精度を高め，適当な操作で孔のずれについて，ある程度の余裕を認めることができるようになる。

電子計算機で使うことのほかに，紙テープは，特殊な電動タイプライタとつないで一般的に用いられている。必要な文字をタイプすると，テープには自動的にその文字が穿孔される。もしこれが再製したい文書だとすると，そのテープを用いることによってタイプライタを自動的に動かすことができる。テープは何回でも使えるので同じ文章を何回も作ることができる。

12-4 磁気テープ

磁気テープ装置（図 12-8）は，最も広く使われ，また大きなデータ記憶装置である。2 千万ビットもの情報を 1 本のリールに蓄えることができる。平均呼出し時間（特定のデータの要求があったときから，そのデータが読み出されるまでの時間）は秒の単位ではあるが，いまだに磁気テープは動作の速い装置の 1 つである。はるかに短い呼出し時間で動作する磁気ディスクについては，12-5 節で述べる。磁気テープはもちろん記憶装置ではあるが，呼出し時間が長いので，算術演算装置と直接関連しては使われない。動作速度の増大（呼出し時間の短縮）のための改良は行なわれているが，普通の呼出し時間は非常に長いので，電子計算機の記憶装置の中心としては用いられていない。磁気テープの上に書かれる 2 進データは，2 方向のいずれかに磁化された点として蓄えられている。通常の 1/2 インチ幅のテープに，7 本のトラックがある。テープの長さは 50 フィートから 2400 フィートにおよぶ。データはブロックにして蓄えられる。データは順に読取り（書込み）または消すことができる。データを消し，新しいデータに置きかえることができることは，磁気テープの重要な特長である。たとえば，銀行の記録は，その目的で使うときには，絶えず，更新することができる。一般に用いられるたいていの他の入出力記憶媒体は永久的で，新しいデータを入れるときには，捨てなくてはならない。

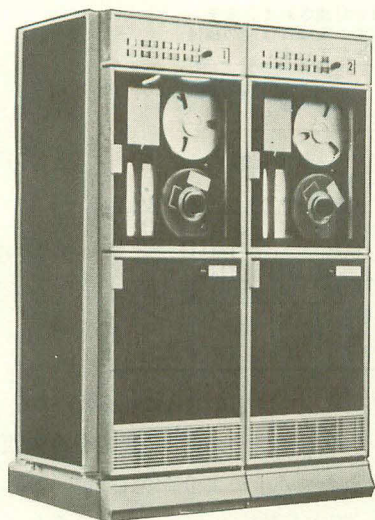
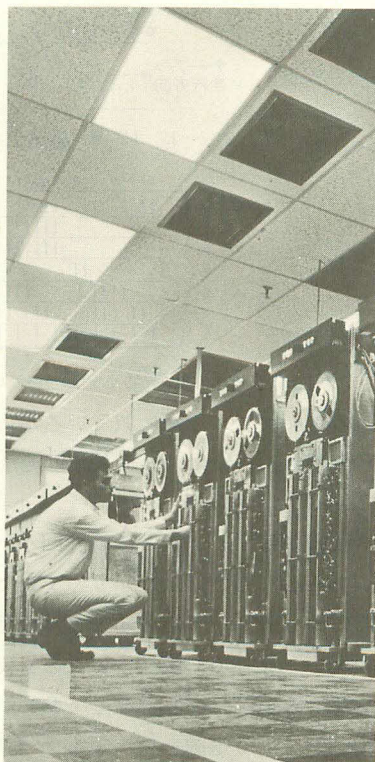
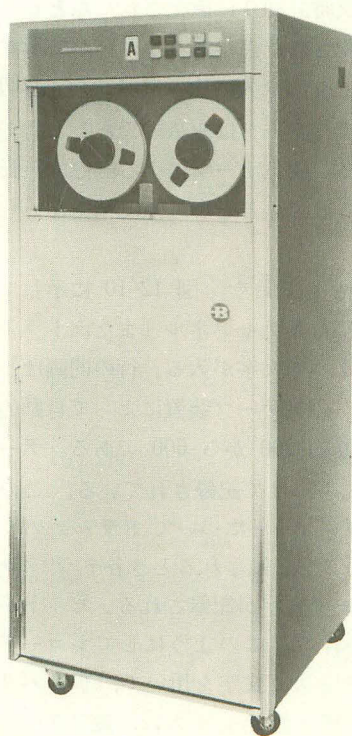
図 12-9 に磁気テープ装置の読取り・書込みヘッドを示してある。それは，家

(a) GE

(b) Minneapolis-Honeywell

(c) Burroughs Corp.

図 12-8 磁気テープ装置

図 12-9 磁気テープでのデータの記憶

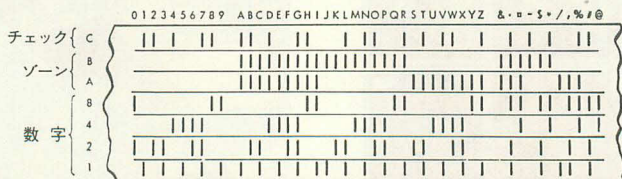



図 12-10 英数字（アルファメリック）コードの表示（IBM 社提供）

庭用のテープレコーダに似ているが，より多くのトラックがあり，音楽や声の複雑な波型のかわりに，データは単純な磁化された「点」により記憶されている。図 12-10 に記号がテープに 7 ビット英数字（英字と数字）コードで蓄えられている状態を示す。ビットは時計回り，または反時計回りに磁化された点として書かれるが，ここではわかりやすいために磁化された点があるかないかで示すことにする。データを消すことはできるけれども，永久にそのデータを（適切な取扱いと保管で）保持することもできるので，データは永久的であると考えられることをはっきり認識しておくべきである。磁気テープによる記憶は装置を動かしている電源を切ってもデータは蓄えられているという点で消えないものと言える。

磁気テープ装置は，入力と出力両方の装置として働く。図 12-10 に示してあるテープコードの形で，データは 7 本の平行なチャンネル（またはトラック）に記録され，各列（テープの幅方向）に 1 つの文字が入る。行の間隔はヘッドの間隔で固定されている。文字の間隔は，磁気テープ装置によって自動的に決められる。文字の密度は，1 インチあたり約 200 から 600 である。テープの文字は，偶数パリティのチェックビットをつけて記録されている。この文字に対するパリティチェックに加えて，各レコードについてトラックパリティチェックも行なわれる。すなわち，データが記録されるとき各トラックのビットは加えられ，ブロックの終りにチェック文字が記録される。ビットが奇数個あるトラックにチェックビットが書かれる。このようにしてレコードを読むとき各文字ごとのチェックビットとチェック文字を用いて，偶数パリティチェックが行なわれる。これらすべてのデータチェックは，磁気テープ装置で行なわれるので，それは電子計算機本体には関係しない。レコードが計

算機に送られるとき，チェック文字は入らない。

磁気テープ駆動機構（図 12-11）は速い動作速度，速い始動と停止，高い読み書き精度が必要なため，家庭の テープ レコーダ よりはるかに進んだものである。駆動モータは絶えずまわっていてキャプスタンすなわち圧力ローラーがテープを動かしたり止めたりするために使われる。モータが速度を上げるにつれてゆっくり加速するのより，この方がはるかに速く速度を上げることができる。テープを動かすときには，駆動キャプスタンをテープに押しつける。テープを止めるときは，駆動キャプスタンをゆるめ，テープを速く止めるため停止キャプスタンを動かす。このような高速の始動と停止をテープを切らないようにして行なうために，各リールに対し，テープのたるみが真空中に作ってある。テープの上の任意の場所のデータが必要になることがあるので，駆動機構はテープを バック スペース（1 レコード　巻き戻す）したり，リールのはじめ

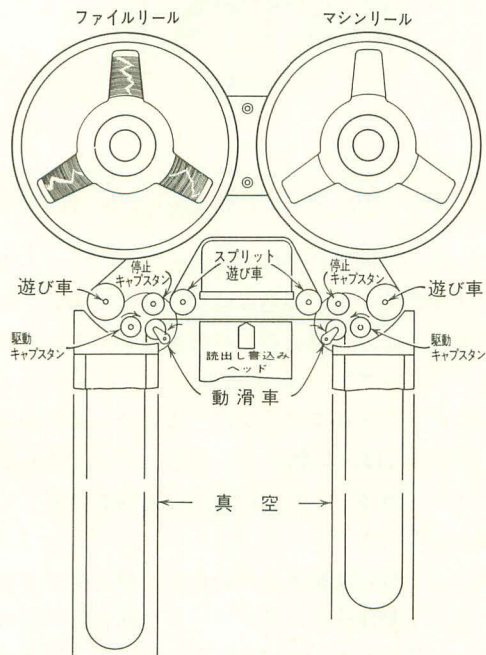



図 12-11　テープ駆動機構（IBM 社提供）

まで巻き戻したりすることができるようになっている。逆方向（データを書いた方向と逆方向）に読めるテープ駆動装置もある。

書込みヘッドにそってテープが動くと，磁束のパルスがテープの表面の点を

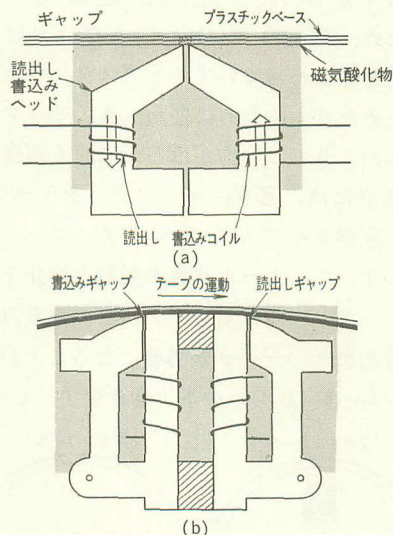



図 12-12 (a) 1ギャップ 読出し・書込みヘッド
(b) 2ギャップ 読出し・書込みヘッド

磁化する。7 つのヘッドは同時に動作する。テープはヘッド上を1秒 100 インチ，またはそれ以上で動くから，書込み時間は非常に短い。そのため，テープは実際には静止しているように見える。一般的なヘッドとして，2 つの型すなわち1ギャップと2ギャップヘッドがある。1 ギャップヘッド（図 12-12a）は，読み書き両方に使うことができるが，一度には一方しかできない。2 ギャップヘッド（図 12-12b）は，1 個ビットを書いたら，それがヘッドの下にある間に読み出すことができる。これは前に述べた検査（正しく書けたか）をするために使用すると便利である。

レコードの大きさは一般に制限されていない。数文字でもいいし，数千文字になってもいい。書込み動作中に，各レコードの間または後に一定長のギャップ（約3/4インチ）が作られる（図 12-13)。このように各レコードの間に**レコード間ギャップ**（IRG, inter-record-gap）があるが，この空白部分があることによって，レコードごとにテープを始動させたり，停止させたりすることができる。

要約すると，磁気テープは，1 インチに 200 から 600 文字，すなわち 2400

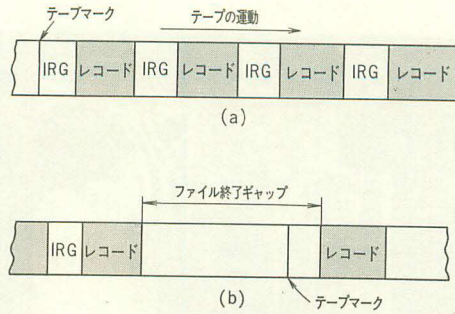



図 12-13 磁気テープのレコード間ギャップ
(a) ファイル終りのテープマーク
(b) ファイル間 (end-of-file) ギャップ

フィートのテープに 2, 3 百万文字のデータを蓄えることができる。データはレコード単位に，またはいくつかのレコードをまとめてブロックにして，順に記憶し，また消すこともできる。記憶装置全体としては高価であるが，1 ビットあたりにすると安い。磁気テープ装置は大きい記憶容量を持っているが，呼出し時間は長い。

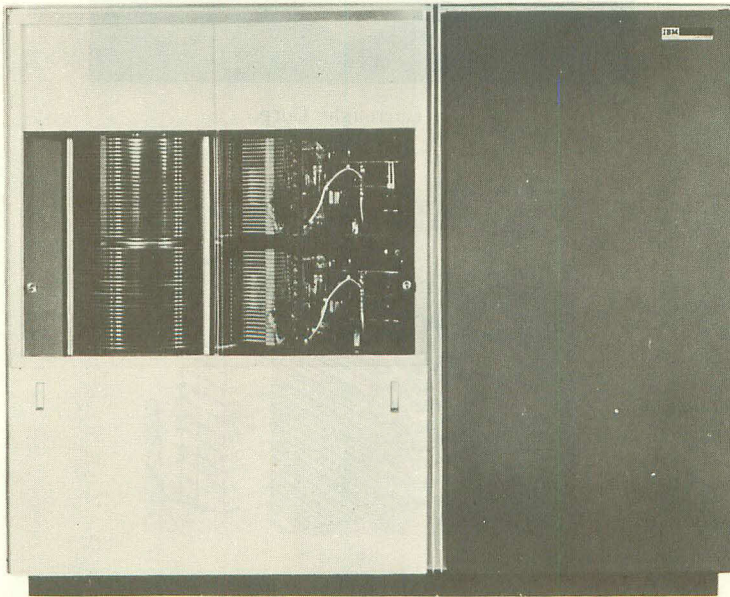
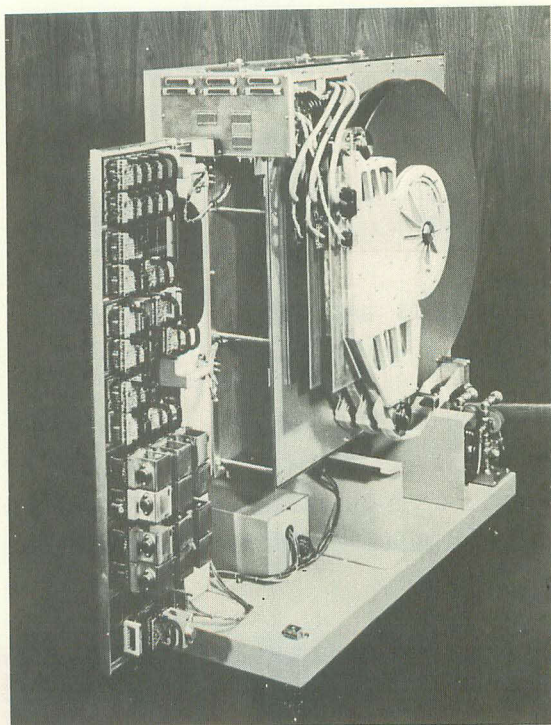
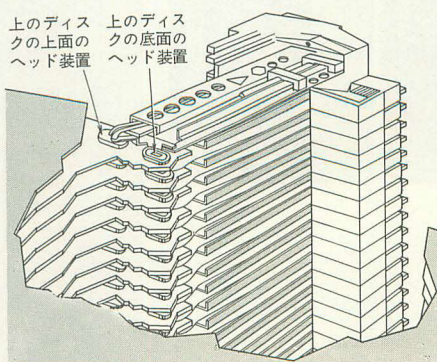



図 12-14 磁気ディスク駆動装置 IBM 1301

(a)　Burroughs Corp.

(b)

(b)　IBM 1301　ヘッド構成

図 12-15　磁気ディクス装置の詳細

12-5 磁気ディスク

磁気ディスク記憶装置は，1 つのユニットとしては最大のデータ記憶装置である（図 12-14）。2, 3 百万文字がディスクの表面に蓄えられる。レコードを録音するのと似た方法で蓄えられるが，溝を切るのではなく，記録は磁性体の表面に行なわれる。また，データを読み出すのに，針は使わないで，可動アームにつけた磁気ピックアップを用いる。データは一般に何枚ものディスクに書かれていて，ディスクごとに読取りヘッドがある。表と裏の両面とも，いつでも読むことができる（図 12-15）。

記録するための各面には，普通 100 から 250 本のトラックがあり，データはトラックに蓄えられる。ディスクの直径は 1 ないし 5 フィートである。ディスクは 1 分間約 1500 回転する。ディスクだけを考えると，平均呼出し時間は約 50 ミリ秒である。しかし，1 つの面に普通，ヘッドは 1 つしかなく，多くのトラックがあるので，読取りアームが指定されたトラックに行くまでの時間を加えると，全体で平均呼出し時間は約 250 ミリ秒になる。これは，磁気ドラムに比べると長い時間のようだが，磁気テープの秒から分と比べればはるかに短い。

データを蓄えるために，NRZ* 記録方式が使われる。呼び出すには，正しいヘッドを選び，それを，必要とするトラックに動かすだけだから，ランダム アクセス と考えられる。磁気テープのように，必要なデータが現われるまですべてを読む必要はない。しかしながら，磁気コア記憶装置の呼出し時間とは比較にならない。データは消すことができ，記録は装置を止めても保存できる。また，多くのディスク駆動装置は，ディスク パック（一定数のディスクから成っている）をとりはずして，別の新しいものと交換することが簡単にできるようになっている。

12-6 高速印刷機

高速印刷機（図 12-16）は，永久的な目に見える記録を作るため，最も一般的に電子計算機で用いられる装置の 1 つである。タイプライタは一度に 1 文字しか書かないが，高速印刷機は一度に 1 行印刷する。そのため，なおいっそう

---

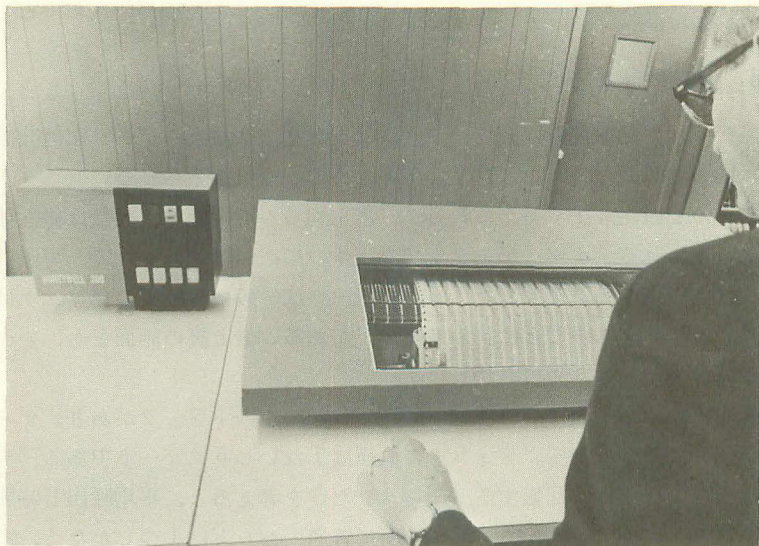
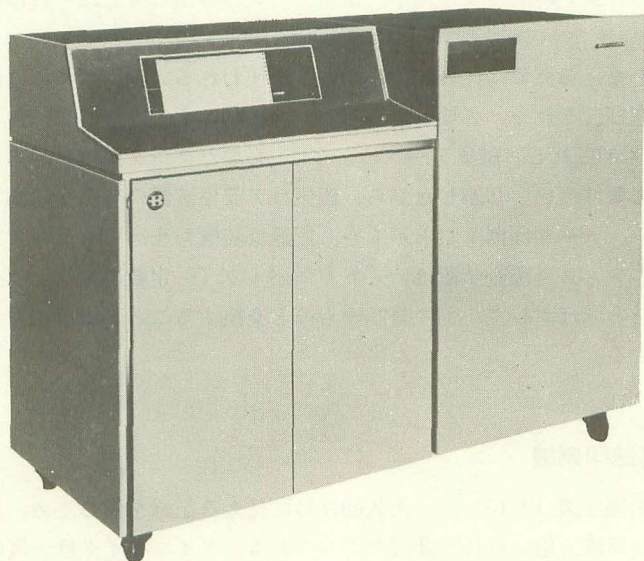
*〔訳注〕 11-3節 磁気ドラム記憶装置の項参照。

(a)　Minneapolis-Honeywell

(b)　Burroughs Corp.

図 12-16　高速印刷装置

速くなっている。また，洗練された機構により，高速の動作ができるようになっている。機械的印刷機と電子的印刷機とがあるが，前者が現在一般に用いられているものなので，それについて詳しく述べる。

印刷機の1つの型として多数の活字輪 (print wheel) または，活字棒 (print bar) を持ったものがある。1行印刷するのに必要な文字の数だけの活字輪または活字棒がある。各活字輪には，英数字と特殊文字が一緒に入っている（図12-17)。各行を印刷する際，活字輪は必要な文字のところまで回転しなくては

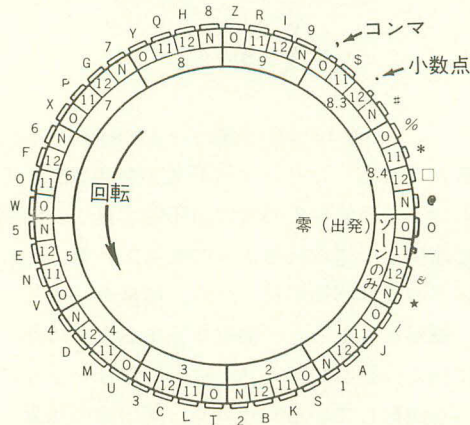



図 12-17 活字輪

ならない。すべての活字輪が正しい位置につくと，機械的に駆動されるハンマー（1文字に1つずつ）が動き，リボンを打ち，その行のすべての文字を一度に印刷する。各行を印刷する際，ある活字輪は他より少ししか回転しないかもしれない。活字輪の動作は始動と停止の激しい反復である。1行印刷すると，活字輪は再び選ばれた場所まで回転を始める。1行印刷するたびに，計算機は印刷装置に新しいデータを送る。1行印刷する時間は計算機のデータ伝送速度と比べて非常に遅いので電子計算機がどのように動作するかにより，直列でも並列でも1行分の情報を送ることができる。1行印刷するごとに紙とリボンはわずかに動き，活字輪は新しい位置まで回転して離れていたハンマーを次の行の印刷のために動かす。このような印刷装置では，印刷速度は普通1分間150行位である。

印刷機の別の型として，回転するドラムの周囲に活字の植っているものがある。ドラムには，1行120文字入る（図12-18)。1行の120文字全体に対する命令は一度に印刷装置に送られる。ドラムの各行が固定した場所にあるハン

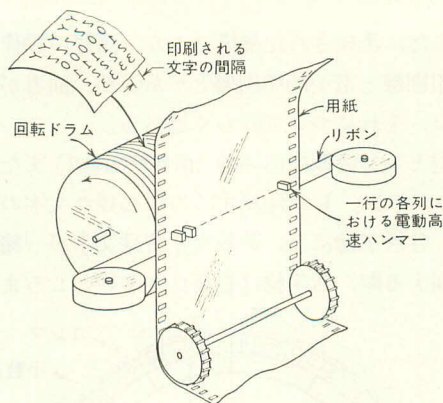



図 12-18 回転ドラム印刷機

マーを通りすぎるとき，個々のハンマーが文字を選び出して印刷する。ドラムが1回転するうちに1行のすべての文字が印刷される。印刷用紙には両側に送り穴がついていて連続して送られるようになっている。そして一定の間隔でミシン目が入っているので，印刷が終ったら，機械を止めないで切り離すことができる。一般に，紙がなくなると，機械は止まるようになっている。機械はこのことを計算機に伝え，新しいデータを送ってこないようにする。動作が連続的（ドラムはいつも回転している）なので，活字輪の始動・停止動作よりもはるかに速く動作する。ドラム型の印刷機は1分間に 200～300 行から 1000 行ぐらい印刷する。また 64 ないし 120 種の文字を印刷できる。

第3の機械的印刷機は連続的に回転する文字ベルト（図 12-19）を使う。ベルト上の必要な文字がハンマーの下にきたとき個々のハンマーが働く。ベルト

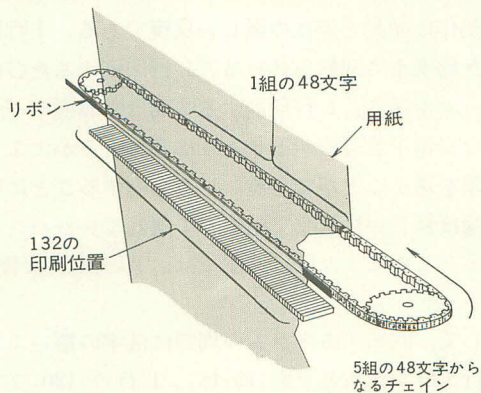



図 12-19 ベルト印刷機

には必要な文字がハンマーの下にくるまでの時間を短縮するためにいくつもの文字の組が備えられている。現在，この型の印刷機では1分間1200行（1秒間20行）以上印刷できる。単に1回のハンマー打ちと紙送り動作をするということではないので，動作を同期させるのは重要である。各ハンマーは，正しい文字がそのハンマーの下にきたとき働く。すべてのハンマーが一度打ったら紙は1行送られ，この動作が繰り返される。この速度は非常に速いように思えるが（実際そうである），計算機の中では，データは何百倍も速く作り出されている。

静電式印刷機では紙とハンマーとの物理的な接触を必要としない。そのかわり特殊な紙を特定のパターンの電気の放電にさらし，必要な文字を作る。格子状マトリックスを使って（電光ニュースのように），特定の文字に対し特定の格子点が選ばれ，文字を"印刷"するため荷電される（図 12-20）。

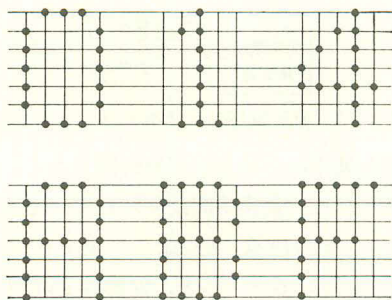



図 12-20 格子文字の作り方

電子計算機や使っているプログラムによるが，1 台以上の印刷装置を使うことがある。そしてそれらへのデータは電子計算機記憶装置から直接か，または磁気テープ装置のような他の入出力装置を経て送られる。小さい電子計算機では，この装置は計算機の記憶装置から直接動かされ，その間他の動作はなにもできない。大型電子計算機では計算機を絶えず動かしていたいため，データは磁気テープ，磁気ディスクパック，または磁気ドラムにだけ送られる。これらの高速入出力装置からデータは穿孔カード，穿孔紙テープ，または高速印刷機に送られる。印刷の制御は，電子計算機から通常2,3のコマンド命令の形で出されるが，その後，操作は入出力装置に引き継がれ，電子計算機はプログラムを続けるために解放される。

12-7 アナログ-ディジタル（A/D）変換

直流電圧は等価なディジタル量にいろいろな方法で変換することができる。基本的に言って，これらの方法では未知の直流電圧を比較するために，いろいろな電圧を発生させる必要がある。既知の電圧を順に変化させていき，直流電圧比較器により既知の電圧と未知の電圧が等しいことがわかるまで，ディジタル カウント を進める。等しくなったらカウントは止まり，そのときのディジタル数は未知の電圧に比例している。図 12-21 にこの操作を簡単に示してある。

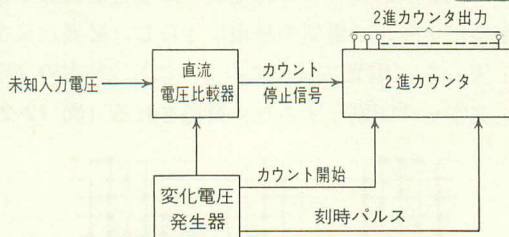



図 12-21　直流電圧から2進カウントへの変換

変換での2つの重要な要素は，精度と変換時間である。精度は第一に比較器回路の精度により決められる。変換時間は使われている刻時周波数とカウントの最大数（すなわちカウンタの段数）とによって決まる。たとえば，1.024 MHz の刻時周波数で 10 段使っているとすると，変換時間は 1/1.024 マイクロ秒×1024 すなわち 1000 マイクロ秒 である。変換をするのに 1000 マイクロ秒 すなわち 1ミリ秒かかるので，毎秒 1000 回の変換が行なえる。より遅い刻時周波数，またはより大きいカウントでは，1 秒あたりの変換の数は少なくなる。もう1つの重要な要素は，変換の解像力である。精度は実際の電圧と，どれくらい近いかを表わすが，解像力は2つの電圧をどのくらい細かく見分けられるかということを表わす。＋1% の精度での 100V は，実際は 99V から 101V のことでそれ以上詳しくはわからない。100mV の解像力というのは 0.1V 以上離れた電圧を見分けることができるということであり，10.6V と 10.65V（50mV の違い）は同じに現われることになる。なぜなら，それは比較器の解像力以下しか違わないからである。しばしば，達成しうる精度より高い解像力で動作させられていることを知らないでいることがある。10.1V と 10.2V とを見分けることができても，実際の電圧は 10.5V であったかもしれない。この場合の精度は解像力に劣っている。他方，高精度であるが低い解像力という

のもよくない。というのは，実際の電圧と非常に近い値が得られるが，わずかな差は判別できない。解像力が 1V で，精度が 1% の場合，10V の信号に対して，0.1V まで正確なのに，9.2V と 9.8V を判別することができない。解像力と精度とを同じ程度に良くすることが重要である。精度と解像力のこの区別は，時々混乱することがあるので，他の分野の仕事においても同様に気にとめておかなければいけない。

刻時周波数と電圧変化は変換のために同期させる必要がある。たとえば，10 mV の解像力では 10mV の電圧変化があるたびにカウンタを1ステップ進める必要がある。このようにすると，10mV の差のあるものに対しては，異なった2進カウントが得られる。一般的な方法の1つとして刻時周波数の変化とともに直流電圧を変化させるために**梯子回路** (ladder network) を用いる方法がある。梯子回路は，2進カウンタによって駆動される電圧解読回路である。したがって，カウンタの各ステップは異なった電圧に変換される。0V からスタートして，直流電圧は，カウントが増すにつれて階段状に増加する。刻時信号がカウンタを駆動しているので，各ステップごとにカウントされ，電圧は1段ずつ増加する。この動作をするために使う回路のブロック図を図 12-22 に示す。

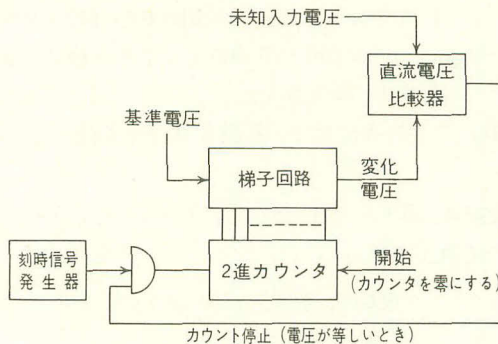



図 12-22 梯子回路を使う直流電圧変換

刻時信号発生器は，必要とする周波数で動作する無安定マルチバイブレータである。直流電圧比較器が2つの電圧(梯子回路の出力電圧と未知の入力電圧)が等しいことを示すまで，刻時信号はカウンタを進める役目をする。基準電圧は，最大変換電圧を指示するために使われる。カウントが進むにつれて，基準電圧の1部が増加しながら梯子回路の出力に現われる。カウンタがフルカウント (すべて 1) のとき出力電圧は基準電圧の値になる。カウンタの段数が変換の解像力を決定する。10 段カウンタでは 1024 分の 1，すなわちおよそ 1000

分の 1 (0.1%) を区別できる。基準電圧が 10V なら，(10/1000)V すなわち 10mV の解像力である。多くの段を使えば高い解像力が得られる。原理的にはたくさんのカウンタの段を使うことによって，解像力をいくらでもよくできる。しかし，変換精度（主に比較器によって決まる）または変換時間（刻時周波数によって決まる）は考えに入れていない。精度は電圧の変化によるので，梯子回路もまた，精度に影響を与える。

例　12-1　8段カウンタの梯子回路を使うと，解像力（%）はどうなるか。

解答：8段で $2^8=256$。256 分の 1，約 $1/250\times100\%$ すなわち 0.4%。

例　12-2　8段カウンタの梯子回路を使うと，10V の基準電圧のとき，解像力（電圧で）はどうなるか。

解答：例 12-1 で，解像力が約 0.4% であることがわかっているから，

$$解像力=0.4\%\times10\text{V}=\frac{0.4}{100}(10)\text{V}=0.04\text{V}=40\text{mV}.$$

例　12-3　10 kHz の刻時周波数で，8 段カウンタの最大の変換時間はいくらか。

解答：8 段では，全部で 2^8 すなわち 256 カウントある。そして刻時周波数 10 kcps であるから，1 カウントあたり $1/(10\times10^3)$(秒)$=100$（マイクロ秒）かかる。必要な全時間は $100\times256=25,600$（マイクロ秒）である。すなわち，1回の変換について 25.6ミリ秒（最大）。

例　12-4　10 kHz の刻時周波数で，8 段カウンタを使うと，1 秒にいくつ変換ができるか。

解答：1回の変換に 25.6ミリ秒 だから，1 秒に $1/(25\times10^{-3})$ すなわち $1000/25=40$ 回の変換ができる。

問題 12-1　次のカウンタの段数で，解像力（%）はどうなるか。

(a)　12　　(b)　6　　(c)　9　　(d)　11

問題 12-2　問題 12-1 について，次の基準電圧を使ったときの解像力（電圧で）を計算しなさい。

(a)　10V　　(b)　1.6V　　(c)　220mV

問題 12-3　問題 12-1 について，次の刻時周波数の場合の変換時間と1秒あたりの変換回数を計算しなさい。

(a)　150kHz　　(b)　820kHz　　(c)　1.6MHz

問題 12-4　基準電圧 41V で，約 10mV の解像力を得るためには，何段のカウンタが必要か。また，1 秒に約 250 回変換をするためには，刻時周波数はどうすればよいか。

問題 12-5　刻時周波数 100kHz, 解像力約 0.2% としたとき，1 秒にいくつ変換ができるか。

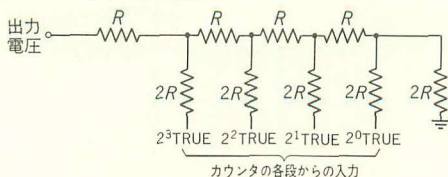



図 12-23　梯子回路

梯子回路は，図 12-23 のように 2 種の異なる値を持つ抵抗をつないでできている。R と $2R$ の値は，たとえば，1kΩ と 2kΩ である。図に示してあるように，各カウンタ段の TRUE 出力は回路の特定の抵抗に結ばれている。これらの入力点での電圧は +10V と 0V(1 と 0) で，出力電圧はこれらが梯子回路のどこに加えられたかによって決まる。どのようにして梯子回路の出力電圧が得られるか，簡単な例について見てみよう。

すべての入力が 0V（入力電圧なし）なら，出力は 0V である。すなわち 0000 のカウントに 0V の出力電圧が対応する。計算しやすいように論理の 1 を +16V とする。2^3 段の入力だけが +16V であるとき出力電圧はどうなるだろうか。2^3 段に +16V がかかり，それ以外の入力はすべて零の場合を図 12-24 に示してある。

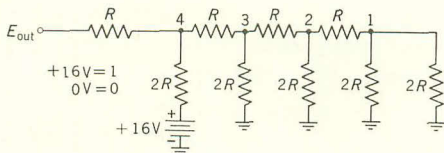



図 12-24　入力 1000 の梯子回路

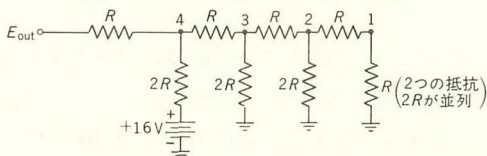



図 12-25　入力 1000 の簡略化された梯子回路

右側から見ていくと，$2R$ が $2R$ と並列になっていて R と等しい（図 12-25)。この R は他の R と直列になっているので，1 つの $2R$ にできる（図 12-26)。再び 2 つの $2R$ が並列なので，等価な R で置きかえる。この R を直列の R に結んであるから，点 3 から右は再び $2R$ になっている。これを繰り返して，$2R$ が点 4 の右側にあることがわかる。この整理された回路を図 12-27 に示す。一般的規則として結合点 1, 2, 3, 4 の右を見ると，抵抗はいつも $2R$ である。結合点 4 の電圧を計算すると $2R$ と $2R$ の電圧分割器を通して

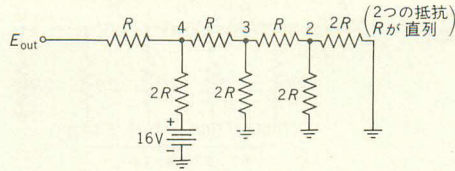



図 12-26　入力 1000 の簡略化された梯子回路

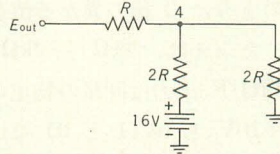



図 12-27　入力 1000 の簡略化された梯子回路

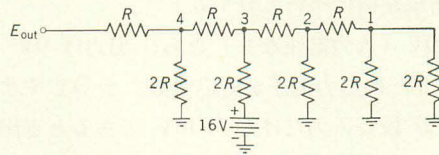



図 12-28　入力 0100 に対する梯子回路

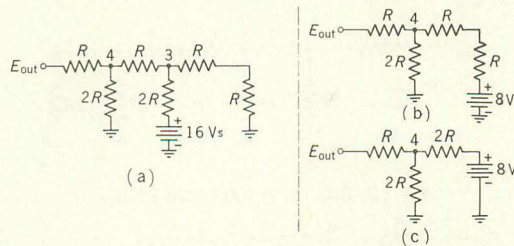



図 12-29　入力 0100 に対する簡略化された梯子回路

16V がかかるので $E_{out}=8\mathrm{V}$ である。最大のカウントの 16(2^4) に対して，カウントが 8 のときは電圧はフルスケールのときの 8/16 すなわち半分である。

入力カウント 0100 について考えてみよう。この入力についての回路を図 12-28 に示す。右側の抵抗はいつも $2R$ なので，回路は図 12-29 のように整理される。

テブナン (Thevenin) の定理*を使って，結合点 3 の電圧を計算すると，図

*〔訳注〕 電源を含んだ回路網の任意の端子から見たインピーダンス Z_0 と，その端子を開放したとき，その端子に現われる電圧 V がわかれば，この電源を含む回路網の等価回路として，電圧が V なる電圧源と，Z_0 との直列回路が得られる。この回路にインピーダンス Z_L を接続すれば，Z_L に流れる電流 I は，$I=V/(Z_0+Z_L)$ で与えられる。これを日本では鳳-テブナンの定理と言っている。

12-29(b) のように，R と直列な +8V となる。図 12-29(c) は 2 つの直列抵抗を加えてさらに単純化したものである。結合点 4 の電圧は電圧分割の規則を使って +4V と計算できる。16 の可能なカウントのうちカウント 0100 すなわち $4\left(\frac{4}{16}=\frac{1}{4}\right)$ では全電圧の $\frac{1}{4}$，すなわち 4V の出力電圧が得られる $\left(\frac{1}{4}(16V)=4V\right)$。回路によって作られた，出力電圧と全電圧の比は，2 進カウントと全カウントの比率と同じである。さらに進めていけば，カウント 0010 では出力 2V，カウント 0001 で出力 1V が得られるはずである。重ね合せの原理により，回路の中に 1 つ以上の電圧源（この場合，1 つ以上の 1 がある）があっても計算することができるので，16 通りの入力に対して，出力電圧を求めることができる。4 段カウンタを使い，論理の 1 として +16V，論理の 0 として 0V の場合について，入力カウントと出力電圧の関係を表 12-2 に示してある。

表 12-2 4 段梯子回路のカウントと出力電圧との関係

入力カウント	出力電圧(ボルト)
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

回路をより一般的にながめると，出力から一番遠い入力電圧には，2^0/全カウントの重みがつけられていることがわかる。そして，その隣はその 2 倍の重みを持ち，以下同様にして出力に一番近いところは $\frac{1}{2}$ の重みを持つ。たとえば，10 段使ったとき，1 番低い値の段は，$2^0/2^{10}$，すなわち 1/1024，最大は

$2^9/2^{10}$，すなわち 512/1024=1/2。論理の 1 を 10.24V と定めると，0000000001 に対する出力電圧は 1/1024(10.24)=10mV，1000000000 では，512/1024(10.24)=5.12V，そして 1000000001 は 5.120+0.010 すなわち 5.13V となる。

例　12-5　次の 2 進カウントに対する 5 段梯子回路の出力はどうなるか。ただし，+6.4V を 1，0V を 0 とする。

(a)　10000　　(b)　00001　　(c)　01000　　(d)　01101　　(e)　10010

解答：

(a)　$2^5=32$；10000 は 16 だから，10000 に対しては $\frac{16}{32}(6.4\mathrm{V})$ すなわち 3.2V。

(b)　00001=1 だから，$\frac{1}{32}(6.4)=0.2(\mathrm{V})$

(c)　01000=8 だから，$\frac{8}{32}(6.4)=1.6(\mathrm{V})$

(d)　01101 に対しては，$\left(\frac{0}{32}+\frac{8}{32}+\frac{4}{32}+\frac{0}{32}+\frac{1}{32}\right)\times 6.4$，すなわち $\frac{13}{32}(6.4)=2.6(\mathrm{V})$。

(e)　10010 に対しては，$\left(\frac{16}{32}+\frac{0}{32}+\frac{0}{32}+\frac{2}{32}+\frac{0}{32}\right)\times 6.4$，すなわち $\frac{18}{32}(6.4)=3.6(\mathrm{V})$。

例　12-6　8 段梯子回路を使うとき，次のカウントについての出力電圧はどうなるか。基準電圧は 51.2V とする。

(a)　10110100　　(b)　10011101　　(c)　00011100

解答：

(a)
$$E_{\mathrm{out}}=\frac{2^7+2^5+2^4+2^2}{2^8}(51.2)=\frac{128+32+16+4}{256}(51.2)=\frac{180}{256}(51.2)=180(0.2)=36(\mathrm{V})$$

(b)
$$E_{\mathrm{out}}=\frac{2^7+2^4+2^3+2^2+2^0}{2^8}(51.2)=\frac{128+16+8+4+1}{256}(51.2)=157(0.2)=31.4(\mathrm{V})$$

(c)
$$E_{\mathrm{out}}=\frac{2^4+2^3+2^2}{2^8}(51.2)=\frac{16+8+4}{256}$$

$=28(0.2)=5.6(\mathrm{V})$

問題 12-6 基準電圧 12.8V の6段梯子回路の出力電圧を，次の2進数について求めなさい。

(a) 101010 (b) 111111 (c) 010101 (d) 101101 (e) 100001

問題 12-7 図 12-30 に示す回路の出力電圧はいくらか。

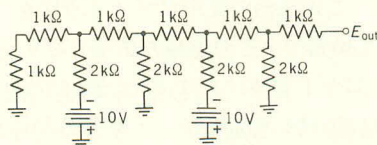



図 12-30 問題 12-7 の梯子回路

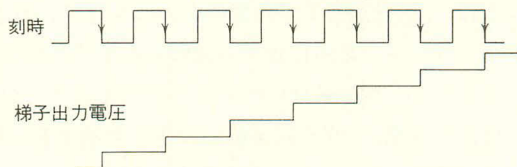



図 12-31 梯子出力電圧（階段波形）とカウンタの刻時信号

梯子回路の出力電圧とカウンタの刻時信号との関係を 図 12-31 に示してあるが，この出力電圧はその形から階段波形（staircase waveform）と呼ばれることもある。

図 12-32 に梯子回路の階段状の電圧が未知の電圧より大きくなったとき，比

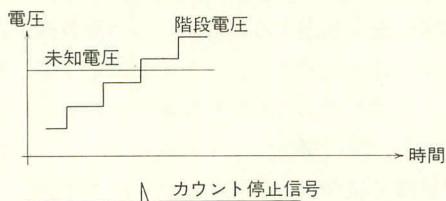
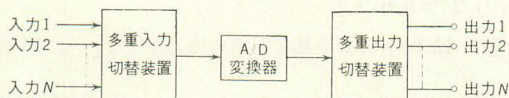



図 12-32 電圧比較とカウント停止信号

較器からカウントを止めるためのカウント停止信号が出される様子を示す。そのときのカウントレジスタの中のカウントは未知の電圧に等価な2進数である。次の変換のためにカウンタが零にされると階段電圧も零に戻る。そして，未知の電圧が変化すれば，再びカウントを進めて行き未知電圧とまた交さする。

1つの A/D 変換器がいくつもの異なったアナログ入力を取り扱うために使われることがしばしばある。1つの入力について変換が終ると，回路はリセットされ，新しい入力が変換器に加えられる。1つの装置で情報の多くのチャンネルを取り扱うために，1つの入力信号から他に切り替える方法は**多重切替え**(mul-

図 12-33 A/D 変換器と多重切替

tiplexing）と呼ばれる。この機構の簡単なブロック図を 図12-33 に示してある。図は1つの A/D 変換器で多くの信号のチャンネルを扱っていることを示す。変換と多重切替え時間が十分速ければ，動作は実時間で行なわれているように見える。変換器は短時間の間，特定のチャンネルの信号だけを見ていて，そこが終ってから他のチャンネルの情報に移って行くが，1 秒間に何回も同じチャンネルに戻ってくるので，あたかも1つの信号をいつも見ているように見える。もし，信号がサンプル（チャンネルに対する離散的な動作）をとる速度に比べて，ゆっくり変化するなら，変換器はいつも1つの信号を見ているように見えるだろう。たとえば，1 秒間に 1000 回変換ができる変換器を，10 個のチャンネルに対して使うと，各チャンネルは，1 秒間 100 回変換される。もし信号が 2Hz の速度でゆっくり変化すると，1/100 秒での変化は，無視しうるほど小さく，変換器が常に信号を見ているように見える。サンプルされたデータに関する研究は非常に複雑であるので，ここでは扱わないが，基本的には変化している信号を研究しているのである。多重切替えそのものは多くの入力を1つのチャンネルに切り替える方法であり，多重切替え装置はサンプルをとる装置である。電子計算機は非常に速く動作するため，1 つの計算機で多くのチャンネルのサンプル値を処理することができる。このことによって，電子計算機は制御操作を行なうのにより有効な道具になるのである。1 チャンネルにつき1台の計算機は経済的でないが，飛行機または宇宙船のシステム全体のたくさんのチャンネルを1つの計算機で扱うなら現実的なことになる。

直流電圧をディジタル数に変換する別の技術として，未知電圧と比較するために一定の割合で変化する電圧掃引（sweep）を使うものがある。図 12-34 に

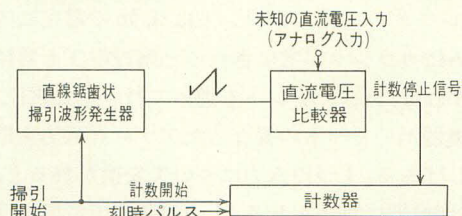



図 12-34 鋸歯状掃引 A/D 変換法

この回路のブロック図を示す。直線鋸歯状掃引波形は一定の割合で増加する直流電圧である。この電圧が未知の直流電圧と等しいか，わずかに高くなると直流電圧比較器により，カウントを止めるための信号が出される。カウントと掃引は，同時にスタートするように同期されている。そして，アナログ入力電圧に等価なディジタル数が得られる。重要なのは(実際にそうするのはむずかしいが)，全掃引間隔の決め方とその間に全カウントが得られるように刻時周波数を正しく調整することである。図 12-35 に，未知の直流電圧に 2,3 サイクルの鋸歯状信号を重ね，その結果として得られるカウント時間間隔を示してある。

もう1つのよく使われる変換は，軸の回転位置に対応する2進カウントを作り出すものである。1 回転する可変抵抗器を考えると，その回転位置には 360 度を対応させることができる。もし，回転の量を可変の信号とすると，電子計算機は回転した量すなわち正確な回転の位置に等価なディジタル数を必要とする。これは飛行機の翼のフラップの回転の度合や制御弁を開けるためのシャフトの回転の量を表わしている。電子計算機でこの情報を計算に用いるためにはディジタル数に直す必要がある。空間が十分にあって設計上可能なら，光学的

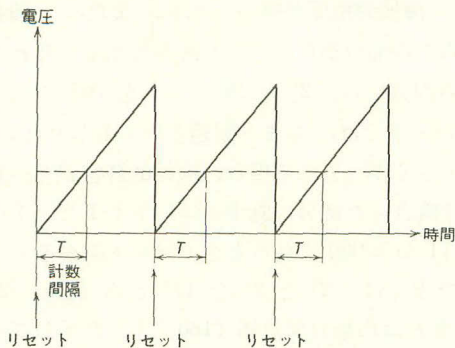



図 12-35 入力と鋸歯状電圧

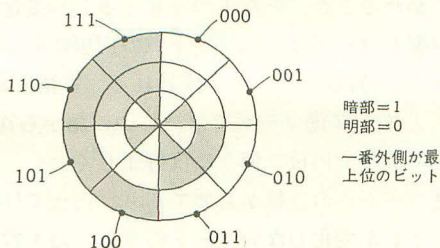



図 12-36 3ビット円板

符号器がしばしば使われる。符号器は透明な部分と不透明な部分のある円板であって，明暗によってコード化されている。図 12-36 に，2 進コードを使った 3 ビット符号器を示す。光の存在を光電池を使って検出すれば，円板の 8 つの異なった扇形を区別することができる。3 ビット使って 360° の円周が 8 つの部分に分けられるので，360°/8 すなわち 45° の角度を持つどの部分にいるかを知ることができる。通常，10 ないし 15 ビット円板が使われるので，$360°/2^{15}$ すなわち $360°/32,768 \cong 1'$ の解像力がある。このように高い解像力を持たせるためには，光学的円板の精密な製作と高度の機械的精度が必要とされる。

例　12-7　10 ビットの円板を使うと，解像力は何度か。

解答：10 ビットでは，解像力は $360°/2^{10}=360°/1024=0.36°$ すなわち，約 $(1/3)°$。

問題　12-8　12 ビットの円板を使うと，解像力は何度か。

問題　12-9　1° よりよい解像力を得るためには，何ビットの円板が必要か。

円板を実際に使用する際，読み取るとき，あいまいになったり，または重複したりすることが大きな問題である。たとえば，2 進コード円板を使えば，000 は 001 の次である。機械的精度が悪かったり，または，調整不良があったりすると，光電池は両方の部分からビットを読み取るかもしれない。すなわち，2^0 ビットを 000 の部分から，2^1 と 2^2 ビットを 001 の部分から読むかもしれない。3 ビット コードでは，あまり問題ないかもしれないが，16 ビットの円板で可能な限り小さく作ってある場合，機械的許容誤差が重要となってくる。重複というのは隣合った部分に対するコードが 1 ビット以上離れているときの問題である。111 から 000 になるとき 1 ビット誤まると，非常に違った語になってしまう。たとえば，2^0 と 2^1 を 111 から読み，2^2 を 000 から読むと 011 となる。それは円板の反対側（180° 先）を示してしまう。多くの場合，180° 先の場所を読むと大破綻をきたす。この問題を正す 1 つの方法は，1 つの扇形から次にかわるとき，ただ 1 つのビットしか変化しないコードを選ぶことである。この型のコードはあるビット数の円板に対していろいろあるが，ほとんどの場合，ある特定のコードが使われる。算術計算のためには，2 進コードが一番よい。しかし 2 進コードでは，1 つの語から次になるのに，1 つ以上のビットが変化するので円板で使うには好ましくない。したがって，円板でのコードから 2 進コードへの変換が必要であり，円板で使うコードとして望ましいのは，1 ビットしか変化しないコードのうちで最も容易に 2 進数に変換できるコードである。このコードはグレイコードである。

全体の操作は機械的な回転位置をディジタル数に変換することであったことを思い出すと，グレイ-2 進変換はこの A/D 変換の主要部分である。グレイ コードは1度に1ビットだけ変化し，循環するコードである。循環するコードでは，コードの語は定まった順で並んでいて，すべての可能な語を使い終ると，次の語は最初に使ったものになっている。このように，このコードの型はすべての語を経て循環する。1 ビットずつ変化するコードのすべてのものが循環するわけではないが，グレイ コードは循環する。表 12-3 に 10 進数の 0-15 について，グレイ コードと2進コードを示しておく。表を見ればわかるように，グレイ コードは，1 度に1ビットしか変化しないが，2 進コードは4ビットすべてかわることもある (7 から8のように)。グレイ コードを2進コードにどのようにして変換するか (またその逆) については，4章で述べてある。

表 12-3　10 進数 0-15 に対するグレイと2進コード

10 進 数	グレイコード	2 進コード
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

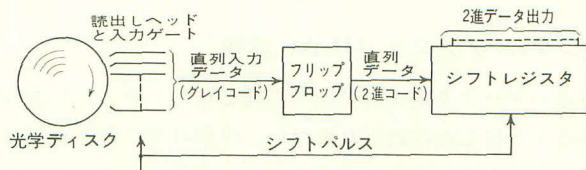



図 12-37　グレイ-2進コード変換器

ここでは実際に回路を作る方法を述べよう。前にも述べたように，グレイ コード は変換しやすいという点で最も望ましいコードである。事実，必要なのは 1つのフリップ フロップ だけである。しかしながらデータを処理するために シフト レジスタ が使われる（図 12-37）。

円板上の明るい部分と暗い部分という形のデータは，光源からの照明を光電池または読取りヘッドで検出することによって読み取られる。光が透明な部分を通ると，光は読取りヘッドを刺激し，不透明な部分にあたると，光は通過しない。読取りパルスは，リング カウンタ から出され，語の各ビットは，その MSB (Most Significant Bit (最上位)) から直列にフリップ フロップ に読み込まれる。直列の グレイ コード のデータは，トリガ入力として入れられて，決まった方向の変化のあるときだけ フリップ フロップ の状態を逆にする。たとえば，NPN 回路では，パルスが高い電圧から低くなるときである。正の論理を考えると，1 から 0 の変化だけが フリップ フロップ を変化させる。もし入力が 0 のままであるか，または 1 になり，そのままでいるときには，出力はかわらない。事実，それは グレイ コード のデータを 2 進コードに変換する方法である。そして，各シフトパルス（読出しパルス）ごとに，フリップ フロップ からの 2 進データはシフトレジスタに読み込まれる。適当な数の読出しステップ（そして シフト ステップ）のあと，シフト レジスタ の中の数は円板の位置（すなわち軸の回転の位置）を示す 2 進コードの語になる。もし シフト レジスタ を バッファ レジスタ または変換装置の一部と考えると，図 12-37 の回路は A/D 変換器全体である（読出しパルスを発生するための論理ゲートを含む）。多くの光学的円板を使うときは，多重切替えを行なえば，1 つの変換回路ですべての入力を処理できる。1 つの円板からの入力を変換したあと，多重切替え装置は読出しパルスを別の円板に切り替え，そこで読み取ったコードを変換する。各変換ごとに，シフト レジスタ の中の語は制御または計算のために電子計算機に読み込まれる。

12-8 ディジタル-アナログ（D/A）変換

ディジタル量のデータを電子計算機で処理したら，その結果を外部のシステムで使用できるようにしなければならない。汎用の電子計算機では，この結果は印字した形で出したり，または穿孔カード，穿孔テープ，または磁気テープに蓄えられる。特殊用途の機械では，得られたディジタル量の答を，外部シス

テムの一部を制御，または動作させるために使用することが多い。そのための信号を作るために，D/A 変換器を使って，ディジタルデータをアナログ量に変換する必要がある。変換は，ディジタル数または語に対してそれと同等な直流電圧を作ることである。そのための方法としては，A/D 変換器とほとんど同じ回路と手法を使う。しばしば同じ装置が使われる。図 12-38 に D/A 変換を行なうために必要な単純化した回路を示してある。

実際には，この回路は A/D 変換回路の一部である。ディジタル数が計算機で

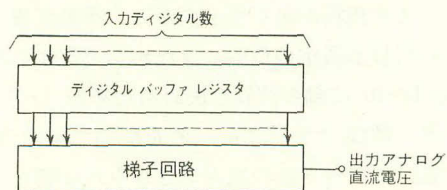



図 12-38 直流電圧，ディジタル-アナログ変換器

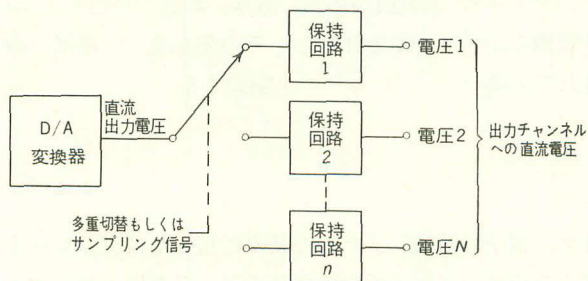



図 12-39 多重切替 D/A 出力直流電圧

求められると，その数は，梯子回路への入力を与えるディジタルバッファレジスタ に入れられる。A/D 変換器について前節で述べたように，梯子回路は与えられたディジタル数に等価な直流電圧を作り出す。多重切替えして多くのチャンネルからのデータを1つの変換装置で処理できるようにすることもできる。その場合，梯子回路を切り替えても，出力電圧を維持するための回路が必要となる。その回路は，“保持 (hold) 回路” と呼ばれ，単に大きい容量のキャパシタと考えればよい。もし，梯子回路で作られた電圧で大きいキャパシタを充電すれば，信号がなくなったあとでも電圧は保持される。各チャンネルにキャパシタを使って，梯子回路は多くの異なった電圧を作り出すことができる。1 秒間に十分な回数各チャンネルに電圧が発生すれば，連続したように見える直流電圧が出力に得られる。図 12-39 に多重切替えと保持回路を示す。

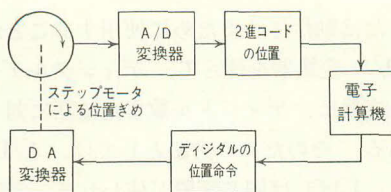



図 12-40　D/A 変換と軸位置符号器

光学的符号器を，特定のディジタル量に対応する位置に動かすためには，他の回路が必要である。まず円板の軸をまわすのに，駆動装置（モータ）が必要である。第2に，いつ円板が所定の位置にきたかを決定するための比較をしなければならない。図 12-40 に軸の位置を決めるための D/A 変換の典型的な回路を示す。ディジタル数は ステップモータ を駆動するために パルスに変換される。各パルスごとに，モータは円板を1きざみだけ動かす。与えられたディジタル数を数えて，円盤は所定のステップ動かされる。円板がどこにあるかを知るために，グレイ-2 進変換が必要である。2 進での位置表示が使えれば，それを電子計算機の示す2進数と比較し，その差を使って漸減計数しながら円板の軸に結ばれている ステップ モータ を駆動する。

要　約

入出力装置は，電子計算機システムの効率に重要な役割を果たす。というのは，それは多くの種類の問題に対して大量のデータを取り扱う能力を持っているからである。いろいろな型の装置はそれぞれ長所と欠点を持っているので，どれも特に優れているということはない。穿孔カードは入力のため広く使われている。1 枚のカードに1つの項目を入れて個々の情報を取り扱えることは便利であり，現在最も実用的な方法である。もしそのカードが学校の生徒を表わすものだとすれば2,3 人の生徒が追加されても，対応する2,3 枚のカードを追加するだけでよい。電子計算機用のプログラムを最初カードにしておくと，あとで2,3 枚のカードを加えたりまたは変更したりして，再びプログラムを実行させるのにも便利である。一方,穿孔テープでは変更を加えるのはめんどうで，新しくテープを作り直さなければならない。紙テープの長所は，安いということといろいろな長さのテープの取扱いが便利なことである。

磁気テープと磁気ディスクは大量のデータの貯蔵のために広く使われている。両方とも大きいプログラムや大量のデータを格納し，電子計算機の速い処

理を可能にしている。銀行や保険会社ではこれらの磁気記憶装置に全顧客の記録を蓄えてある。データは最初カードに穿孔するが，電子計算機で使う前にテープまたはディスクに入れる。そして計算機の演算の結果はまたテープやディスクに出される。

出力データは，穿孔カード，紙テープ，磁気テープ，またはディスクに出される。印刷形式の出力が必要なところには，高速印刷機が広く使われている。1 分間に 1800 行も印刷するので，大量の印刷物が電子計算機から出てくる。タイプライタは主に電子計算機の運転のために，または問題点の発見のために使われる。

特殊用電子計算機では，アナログ-ディジタル，ディジタル-アナログの変換装置が重要である。直流または交流電圧の変換，および軸位置の変換は広く使われていて，この変換を用いることによって，飛行機，ロケット，製造工場などのシステムの操作を制御するのに電子計算機を使用することができる。

問　題

1. 表 12-1 と図 12-3 を使い，図 12-41 のカードを読みなさい。

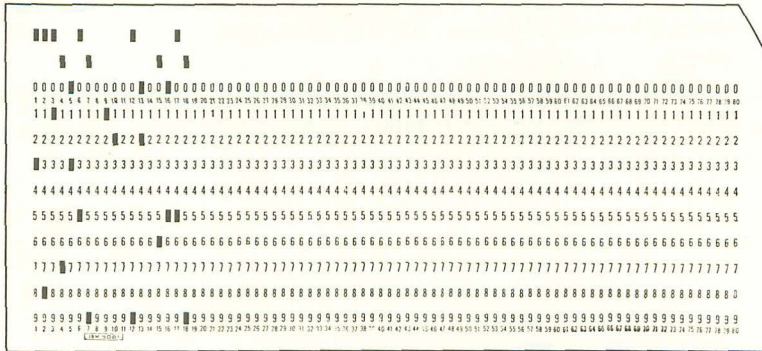



図 12-41　問題 1 の穿孔カード

2. 図 12-7 を使い，図 12-42 の 5 単位の穿孔テープを読みなさい。

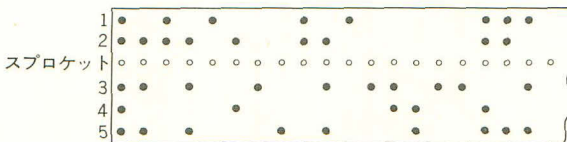



図 12-42　問題 2 の紙テープ

3. 4章の8単位 ASCII コードを使い，図 12-43 の磁気テープを読みなさい。

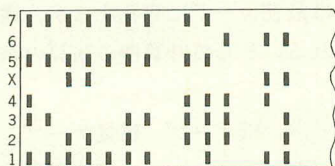



図 12-43 問題3の ASCII コードの文

4. 1つの磁気ディスクに，7枚のディスクがついていて，各ディスク面には 100 トラックあり，1トラックに 16,384 ビットある。このディスク パック の全記憶容量はいくらか（ディスクの両面使えることに注意）。

5. 文字ベルトを使っている高速印刷機では，ベルトに4組の文字がある。ベルトが1分間 100 回転すると，1 分間に約何行印刷できるか。

6. 刻時周波数 512 kHz, 12 段カウンタを使う A/D 変換器の最大変換時間を求めなさい。

7. 梯子回路を使う直流電圧ディジタル変換器で，基準電圧 20V, 12 段の2進カウンタを使うと，解像力はどれだけか。

8. 基準電圧 10V で，解像力 10mV を得るためには，何段のカウンタが必要か。1秒に，約 100 回結果が欲しい場合，刻時周波数はいくつにしたらよいか。

9. (a) 2kΩ と 4kΩ の抵抗を使う4段梯子回路を書きなさい。

(b) 同じ回路で，1011 の入力があった場合の状態を，最小桁を右にして，示しなさい。1の状態を表わすのに 12V, 0 には 0V を使うものとする。

(c) 上の状態での出力電圧を計算しなさい。

(d) その答を，ディジタル カウント と比較しなさい（10000 カウントのうちの 1011）。

10. 問題9を，6 段梯子回路，入力 101011，基準電圧 6.4V とかえて解きなさい。

問 題 解 答

3 章

1. (a) 55 (b) 56 (c) 21 (d) 42 (e) 126
2. (a) 11001 (b) 1000011 (c) 1100011
   (d) 10000111 (e) 100010100
3. (a) 0.625 (b) 0.75 (c) 0.125 (d) 0.375 (e) 0.78125
4. (a) 0.111 (b) 0.0001011111... (c) 0.01011110...
   (d) 0.10001 (e) 0.0111
5. (a) $(37)_{10}$ (b) $(77)_8$ (c) $(85.375)_{10}$ (d) $(167)_8$ (e) $(1161.4)_8$
6. (a) $(155)_8$ (b) $(011111010)_2$ (c) $(56.7)_8$
   (d) $(010111101011)_2$ (e) $(010101.101111)_2$
7. (a) $(1000001)_2$ (b) $(1000110000)_2$ (c) $(410)_8$
   (d) $(4217)_8$ (e) $(1001001)_2$
8. (a) $(10011)_2$ (b) $(2611)_8$ (c) $(246)_8$
   (d) $(10011110)_2$ (e) $(00110100101)_2$
9. (a) $(100001)_2$ (b) $(55717)_8$ (c) $(10100010)_2$
   (d) $(1010)_2$ (e) $(30)_8$

4 章

1. 0010 0101 0111 0011
3. 0011 1001 0010 1000 0111
4. 0101 1010 0100 1011
5. 10 00010 01 00100 10 00001 10 10000
6. 10 10000 10 01000 10 00001
7. 1011011011
8. 100100110
9. 0011 0 0101 0 0111 1 0010 1
10. 0101 1 1000 0 1100 1 1011 0
11. 01010010 01010111
12. 10100001 01001011 10100010 01011101 10100011

5 章

1. A
3. V + UW

4. B

9. U′Y′+V′Y′

6 章

3. (1) A ＝ 1, B ＝ 0, D ＝ 1 の場合，出力は +11.3 V。
   (2) A ＝ 1, B ＝ 0, D ＝ 0 の場合，出力は 0 V。
   (3) A ＝ 1, B ＝ 1, D ＝ 0 の場合，出力は 8 V。
   (4) A ＝ 1, B ＝ 1, D ＝ 1 の場合，出力は +11.3V。

6. 回路の β は 39。$h_{FE}=20$ より大きい。
   トランジスタは飽和しない。

7. $(V_{BE})_{\mathrm{off}}=-1.72\ \mathrm{V}$

7 章

6. $f=1.52\ \mathrm{MHz}$

10 章

1. (a) 00011011　(b) 11011011
2. (a) 001001000　(b) 111000111
3. (a) 0100110　(b) 0001010
4. (a) 1001011　(b) 1111100
5. (a) 0001010　(b) 0101001
6. (a) 11001101　(b) 11110011
7. 11011
8. 10000010
9. 1010
10. 101

11 章

1. X（行番地）に 6 段。Y（列番地）に 6 段。
2. X および Y にそれぞれ 4,5 段，Z に 20 段。
3. Z に 10 段，X および Y にそれぞれ 5 段，6 段。
4. (a) 1.67 ミリ秒。
   (b) 307,200 ビット/秒。
   (c) 3.25 マイクロ秒。
5. 2.5 ミリ秒。
6. (a) 409,600 ビット/秒。
   (b) 409,600 ビット/秒。

12 章

1. CHAPTER 12 IS OVER
2. PAPER TAPE NO. 26
3. MESSAGE NO. 97
4. 22,937,600 ビット。本当は最外側の2面を引くと，19,660,800 ビット。
5. 約 400 行/分。
6. 8 ミリ秒。
7. 約 5 ミリボルト。
8. 10 段，1 MHz
9. (c) 8.25 V　　(d) 8.25 V
10. (c) 4.3 V　　(d) 4.3 V

索　引

A

アキュムレータ (accumulator)　12
アナログ-ディジタル (A/D) 変換　252
AND 関数　68
ASCII コード (American Standards Code for Information Interchange)　62

B

BCD (binary-coded decimal, 2 進化 10 進数)　52
ヴィーチ (Veitch) 図　90
ブール代数 (Boolean algebra)　66

D

ダイオード　105
ダイオードマトリックス　175
電流一致方式　208
電流利得 (current gain)　124
電流増幅率　124
電子式論理ゲート　76
ディジタル-アナログ (D/A) 変換　264
デューティサイクル (duty cycle)　140
ド・モルガンの定理 (De Morgan's theorem)　71

E

エミッタホロワ (emitter follower)　148
EXECUTE　180

F

FETCH　180

G

ゲート素子　74
グレイコード (Gray code)　58, 262

H

ハイブリッド計算機 (hybrid computer)　2
破壊的な読出し　205
薄膜記憶装置　225
半減算器　96
半加算器 (half adder)　90
梯子回路 (ladder network)　253
8 進法　37
方向づけ回路 (steering network)　135
保持回路　265
フェライトシート　225
フィードバック計数器　161
負の論理　109
フリップ フロップ (flip-flop)　134, 154
フローチャート (flowchart)　19

I

インバータ (inverter)　77
位相記録　223
1 安定マルチバイブレータ (monostable multivibrator)　130, 137
1 の補数 (ONE's complement)　44

K

解読マトリックス　209
カルノー図 (Karnaugh map)　84
計数型電子計算機 (digital computer)　1
桁上り (carry)　42
機械語 (machine language)　11
基数 (base)　29
木状結線 (tree connecting)　176
コード (code)　52
コード化 (coding)　52
刻時信号　172
刻時トラック　223

コンパイル (compile)　27
固体素子 (solid-state component)　5
高速印刷機　247

M

無安定マルチバイブレータ (astable multivibrator)　130, 134

N

2 安定マルチバイブレータ (bistable multivibrator)　130, 131
2-5 進コード (biquinary code)　55
2 の補数 (TWO's complement)　44
2 進法　30
2 進化 10 進数 BCD (binary-coded decimal)　52
2 進計数器　154
2 進数 (binary digit)　11
NOR ゲート　81
能動素子 (active device)　105
NRZI 方式　222
NRZ 方式　221

O

オブジェクトプログラム (object program)　27
オペランド (operand)　12
OR 関数　67

P

パリティビット (parity bit)　57
プログラム　5

R

ランダム呼出し　203
RB 方式　221
レコード間ギャップ (IRG, inter record gap)　244
リングカウンタ　173
RZ 方式　220

S

3 あまりコード (excess-three code)　54
算術演算装置　188
制御装置　4, 179
正の論理　109
穿孔カード　230
穿孔紙テープ　236
シフトレジスタ　158
真理値表 (truth table)　69
ソースプログラム (source program)　27
相似型計算機 (analog computer)　1
数体系　29
シュミットトリガ (Schmidt trigger)　130, 147
出発点トラック　223

T

多重切替え (multiplexing)　259
転送率　219
逐次呼出し　203
トンネルダイオード　226
トランジスタインバータゲート　115
トランジスタの特性曲線 (characteristic line)　118
トリガ信号　131
ツイスタ記憶装置　224
直結トランジスタ論理ゲート(DCTL) 126

Y

呼出し (access)　203

Z

全減算器　96
全加算器 (full adder)　90
磁気ディスク　247
磁気コア記憶装置　204
磁気テープ　240
循環桁上り (end arround carry)　46
10進法　29
10進計数器　165

訳 者 略 歴

浦　昭二（うら　しょうじ）

1952年　東京大学工学部応用数学科卒業
1962年　理学博士
現　職　慶応義塾大学工学部教授

主 要 著 書

A.J.ダンカン／品質管理のための統計学（共訳，JUSE 出版社，1954）
T.L.サーティ／オペレーションズ・リサーチの数学的方法（上・下）（共訳，紀伊国屋書店，1960）
M.E.マンデル／動作・時間研究の理論と実際（共訳，紀伊国屋書店，1965）
電子計算機のための数値計算法 I（共著，培風館，1965）
FORTRAN 入門（編，培風館，1966）
F.P.フィッシャー・G.F.スウィンドル／電子計算機プログラミングシステム（訳，培風館，1967）

北川　節（きたがわ　みさお）

1946年　慶応義塾大学工学部電気工学科卒業
1967年　工学博士（慶応義塾大学）
現　職　慶応義塾大学工学部教授

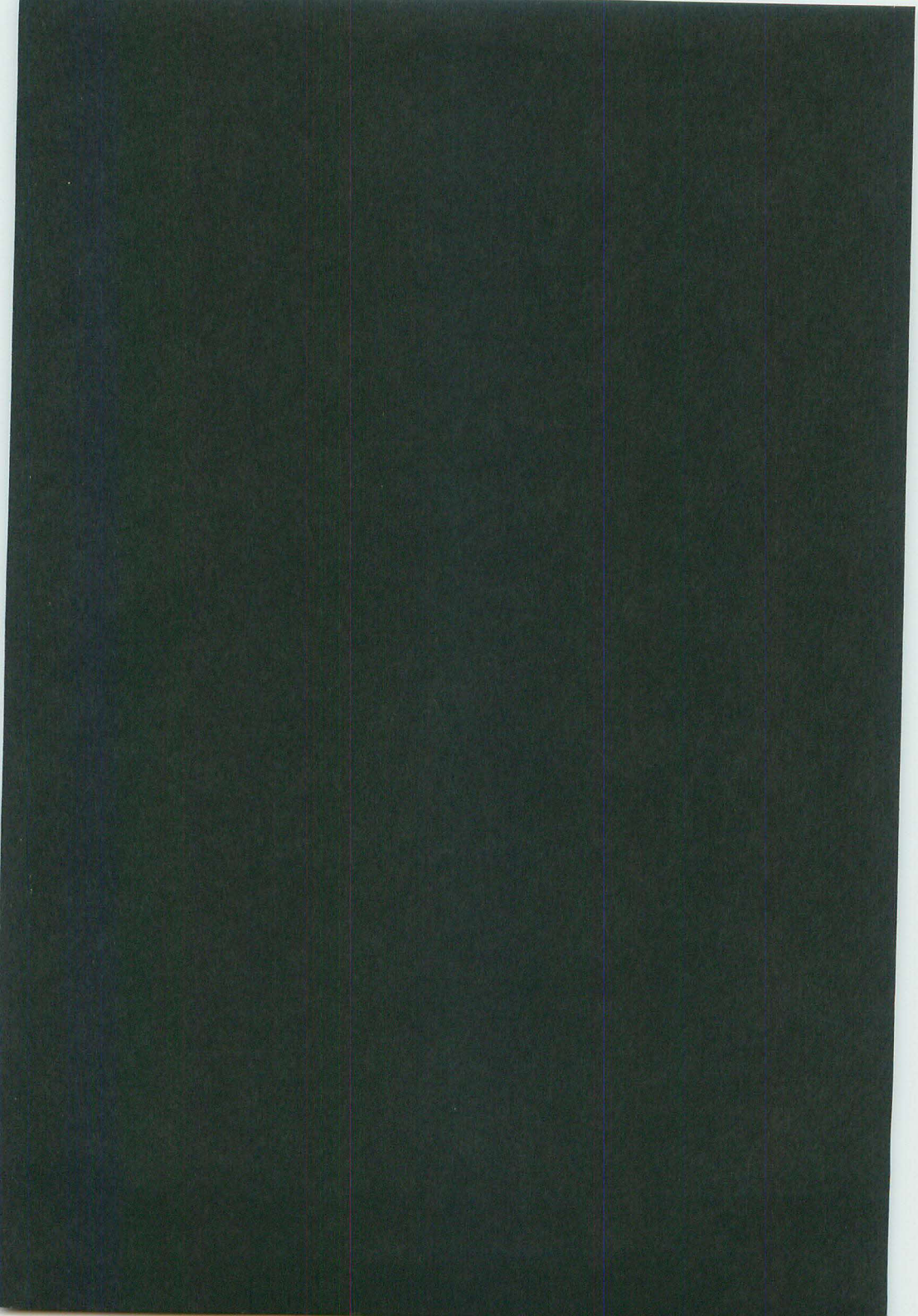
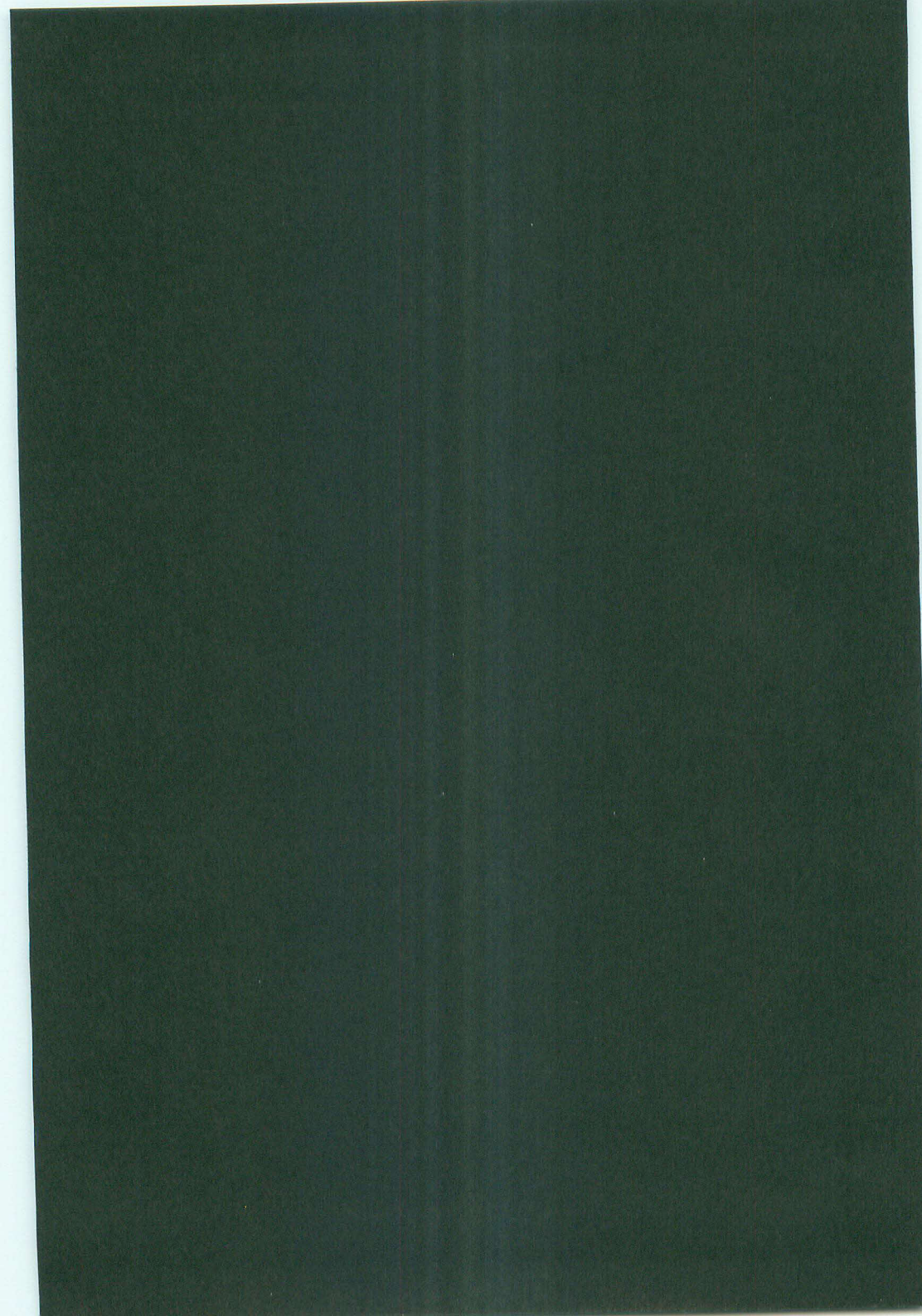
昭和44年1月20日　初版発行
昭和50年10月30日　初版第11刷発行

電子計算機の基礎
—プログラマのためのハードウエア—

著　者　L.ナシェルスキー
訳　者　浦　昭二
　　　　北川　節
発行者　山本健二
発行所　株式会社　培風館
東京都千代田区九段南 4-3-12・郵便番号 102
電話 (03) 262-5256(代表)・振替東京 4-44725

定価 ¥ 1200.

清和印刷・千明社製版・土開製本

訳者の承認をえて検印を省略しました

3055-0703-6955

＊ 物理学

物質の世界＝現代物理へのアプローチ
木下是雄著 / ￥1200

エンジニアのための**物理学**
ポイントン外著 / 斎藤信彦外訳 / ￥1300

応用物理ポケットブック
インゲルスタム外著 / ￥950

基礎課程**物理学演習**全2巻
佐藤瑞穂著 / 上－￥1100　下－￥1000

大学**物理学演習** 佐藤瑞穂著 / ￥1200

演習物理学全2巻
若桑光雄著 / 各￥1200

誤差論 一瀬正巳著 / B6・￥600

力学＝新しい視点にたって
バージャー外著 / 戸田盛和外訳 / ￥1500

熱力学・統計力学 原島鮮著 / ￥1300

＊ 工学

初等力学 森口繁一著 / ￥1300

力学概論改訂版 守屋富次郎外著 / ￥1200

基礎力学演習 杉山隆二著 / ￥1600

工学基礎**流体の力学** 安藤常世著 / ￥1100

材料力学演習全2巻
金沢武外著 / 1－￥1200　2－￥1100

構造力学入門全3巻
高橋武雄著 / 1, 2－￥1700　3－￥2000

工学基礎**熱力学**全2巻 ケスティン著 / 渡部康一外訳 / 1－￥1900　2－￥2200

連続体の力学入門
ファン著 / 大橋義夫外訳 / ￥2600

＊ 固体の力学シリーズ全6巻

1. **粘弾性学**
   フリューゲ著 / 堀幸夫訳 / ￥1600
2. **熱弾性**
   パークス著 / 渥美光訳 / ￥1300
3. **構造安定の原理**
   ツィグラー著 / 砂川恵訳 / ￥1700
4. **構造物のクリープ**
   フルト著 / 村上澄男訳 / ￥1500
5. **弾性平板**
   マルゲール外著 / 玉手統訳 / ￥2200
6. **非線形動的弾性学**
   ブランド著 / 大橋義夫外訳 / ￥1250

機械力学 木内石著 / ￥1900

基礎電気回路1 末武国弘著 / ￥1100

電磁気学 西巻正郎著 / ￥1300

基礎電磁気学演習 佐藤瑞穂著 / ￥950

基礎交流理論演習 佐藤瑞穂著 / ￥1000

空気力学序論 守屋富次郎著 / ￥2400

構造材料の強度と破壊全2巻
テテルマン外著 / 宮本博訳 / 各￥2800

塑性学 ヒル著 / 鷲津久一郎外訳 / ￥2700

塑性加工学全2巻 ジョンソン外著 / 清田堅吉外訳 / 1－￥1800　2－￥1700

マトリックス法による構造力学の解法
マーチン著 / 吉識雅夫監訳 / ￥2300

コンピュータによるマトリックス構造解析法
ジェンキンス著 / 吉田裕訳 / ￥2200

マトリックス構造解析入門
リブスレイ著 / 山田嘉昭外訳 / ￥1800

マトリックス構造解析の基礎理論
シェムニスキー著 / 山田嘉昭外訳 / ￥3200

＊ 新物理学シリーズ　A5変形判 / 208～336頁

2. **電磁気学** 平川浩正著 / ￥1000
7. **磁性** 金森順次郎著 / ￥1300
8. **電気伝導** 阿部龍蔵著 / ￥1300
10. **電子・原子・分子の衝突**
    高柳和夫著 / ￥1100
11. **量子エレクトロニクス入門**
    宅間宏著 / ￥1500
12. **電気力学** 平川浩正著 / ￥1000
15. **ベータ崩壊と弱い相互作用**
    山田勝美外著 / ￥1900
16,17. **物性研究者のための場の量子論Ⅰ,Ⅱ**
    高橋康著 / Ⅰ－￥2200
18. **多体問題** 高野文彦著 / ￥2400
19. **場の量子論** 中西襄著 / ￥3800

別巻　**現代物理用語**
日本物理学会編 / ￥1300

コンピュータによる骨組構造解析
ビューフェ外著 / 成岡昌夫訳 / ￥3200

重みつき残差法と変分原理
フィンレイソン著 / 鷲津久一郎外訳 / ￥5000

固体の力学 / 理論
ファン著 / 大橋義夫外訳 / ￥4000

クリープ強さの理論
オドクヴィスト外著 / 村上澄男訳 / ￥2300

構造動力学の確率論的方法
リン著 / 森大吉郎外訳 / ￥4500

自動組立工学
ブースロイド外著 / 佐田登志夫外訳 / ￥1400

制御システム工学
ドルフ著 / 佐貫亦男訳 / ￥2900

＊ コンピュータによる構造工学講座　全11巻・22分冊

Ⅰ-1. **マトリックス法とコンピュータ** 戸川隼人著 / **有限要素法による構造解析プログラム** 三本木茂夫外著 / ￥2000

Ⅰ-2. **骨組構造解析入門** 信原泰夫外著 / **伝達マトリックス法** 成岡昌夫外著 / ￥2100

Ⅰ-3. **マトリックス法材料力学** 山田嘉昭著 / **エネルギ原理入門** 鷲津久一郎著 / ￥2300

Ⅰ-4. **マトリックス法算法概説** 栖原二郎著 / **マトリックス法振動および応答** 川井忠彦著 / ￥2400

Ⅱ-1. **計算技術および数値計算法** 平野菅保外著 / **骨組構造解析** 成岡昌夫外著 / ￥3200

Ⅱ-2. **塑性・粘弾性** 山田嘉昭著 / **有限要素法のプログラム・デザイン** 信原泰夫外著 / ￥3500

Ⅱ-3. **弾性学の変分原理概論** 鷲津久一郎著 / **有限要素法と破壊力学** 宮本博著 / ￥2400

Ⅱ-4. **動的応答解析** 河島佑男著 / **熱伝導と熱応力** 藤野勉著 / ￥3700

Ⅱ-5. **平板の曲げ理論** 栖原二郎著 / **マトリックス構造解析の誤差論** 山本善之外著 / ￥3400

Ⅱ-6. **シェル構造解析** 川股重也著 / **座屈問題解析** 川井忠彦著 / ￥4900

Ⅱ-7. **薄板構造解析** 川井忠彦外著 / **建築構造物の自動設計と最適設計** 田中尚外著 / ￥3900

＊ 原子炉工学講座　全6巻

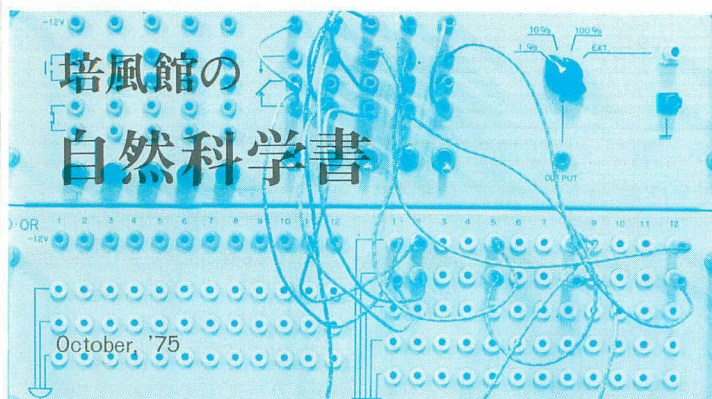
日本原子力研究所　原子炉研修所所長石森富太郎編

1. **原子核工学基礎**　￥2800
2. **放射線防護**　￥2200
3. **原子炉物理**　￥2500
4. **燃・材料**　￥3000
5. **熱工学・構造設計**　￥2200
6. **計装制御**　￥2200

★本掲載書目中には化学・生物学の各部門は掲載してございません。
★最近の資材・印刷・製本料の大幅な値上りによって，あるいは今後もやむをえず若干の定価改正をさせていただく場合も起りうるかと存じますので，何卒あらかじめ御諒承のほどお願い申しあげます。

培風館
東京都千代田区九段南4－3－12（郵便番号102）
振替東京4-44725　電話　東京（262）5256

培風館の自然科学書

October, '75

新刊書より

初等ベクトル解析
群馬大学教授理博道脇義正・熊本大学教授長沢純共著 / A5・P128・￥800

微分積分学要論
横浜国立大学教授理博青木利夫・横浜国立大学教授森川正共著
A5・P304・￥1400

フーリエ変換とその応用
東京大学教授工博近藤次郎著
A5・P272・￥2600

計量経済学序説
R. J. ウォナコット・T. H. ウォナコット共著 / 国府田恒夫・田中一盛共訳
A5・P400・￥2900

コンピュータサイエンス入門
4＝BASIC編
A. I. フォーサイス外著 / 理博浦昭二監訳
土居範久訳 / A5・P152・￥1400

新統計学シリーズ8
統計的予測論
東京大学助教授経博竹内啓著
A5変形・P176・￥2400

整数計画法とネットワーク・フロー
T. C. HU著 / 工博伊理正夫監訳 / Ph. D. 今野浩・Ph. D. 棚橋次彦共訳
A5・P496・￥5800

固体の力学入門全2巻
E. P. ポポフ著 / 名古屋大学教授工博成岡昌夫・大阪大学教授工博浜田実共訳
A5・1－P368　2－P352・各￥1900

電子工学演習改訂版　全2巻
1＝基礎とデバイス　2＝回路と応用
西巻正郎・末武国弘・関口利男共編
A5・1－P224　2－P208・各￥2000

基礎工学における
マトリックス有限要素法
O. C. ツィエンキーヴィッツ著 / 吉識雅夫・山田嘉昭監訳 / A5・P592・￥6000